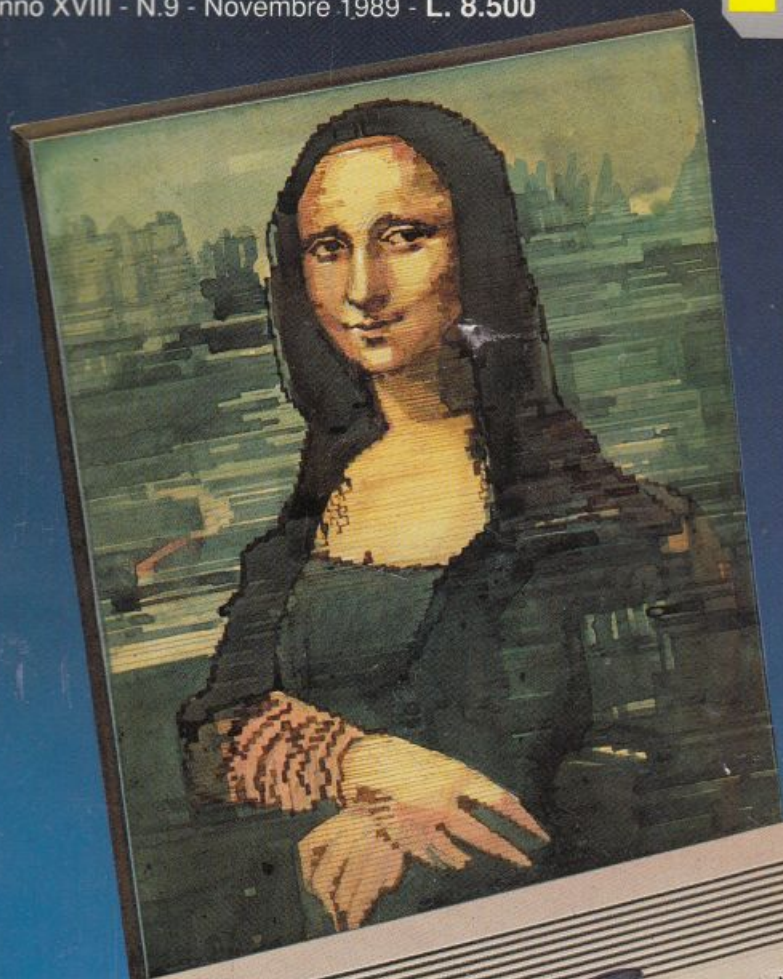
Sped. in Abb. Postale Gr. III/70%

Anno XVIII - N.9 - Novembre 1989 - L. 8.500

Tassa pagata per campione allegato

TUTORIAL
IL C64 TI PRESENTA AL DATORE DI LAVORO

LOGO
AVVENTURA SULL'ISOLA

HARDWARE
PREVENIRE IL MALE DEL DRIVE

AMIGA
A-MAX, L'EMULATORE MACINTOSH

Trasferimento automatico dei programmi da cassetta a disco

FARE GRAFICA!
Hi-res, sprite, finestre, trucchi e superschermi

Gruppo Editoriale JCE

ATARI ST SCREENSHOTS

DRAGON SPIRIT™

Scorch through the skies in the most challenging flight of your life.
Breath fire over hordes of attacking creatures from a prehistoric age.

- A scorching, rip-roaring fight to the death
- Pick up bonuses for mega weapons and extra firepower
- 8 levels of pulse-racing action
- Exhilarating and challenging...
- Have you got the fighting spirit? Have you got Dragon Spirit?

Programmed by: Consult Software Ltd

TENGEN
The New Name in Coin-Op Conversions.

LEADER
DISTRIBUZIONE

Direttore Editoriale
Area Informatica
Marinella Zetti

Direttore responsabile
Paolo Romani

Caporedattore
Fernando Zanini

Responsabile grafico
Desktop Publishing
Adelio Barcella

Impaginazione elettronica
Denise De Matteis

Segretaria di redazione
Alessandra Marini

Collaboratori
Paolo Gussoni, Isa Sestini

RadioELETTRONICA&COMPUTER
Rivista mensile, una copia L. 8.500, numeri arretrati lire 13.000 cadauno.
Pubblicazione mensile registrata presso il Tribunale di Monza n. 679 del 28/11/88.

Fotolito: Bassoli - Milano.
Stampa: GEMM Grafica srl, Paderno Dugnano (MI).
Diffusione: Concessionario esclusivo per l'Italia A.&G. Marco SpA, Via Fortezza 27 - 20126 Milano. Spedizione in abb. post. gruppo III/70.

Abbonamenti: Annuale L. 64.000, estero L. 130.000.

RadioELETTRONICA & COMPUTER è titolare in esclusiva per l'Italia dei testi e dei progetti di Radio Plans e Electronique Pratique, periodici del gruppo Societé Parisienne d'Edition.

Gruppo Editoriale JCE srl
Sede legale, Direzione, Redazione, Amministrazione
Via Ferri 6 20092 Cinisello Balsamo (MI)
Tel. 02/66025.1 -Telex 352376 JCE MIL I -
Telefax 61.27.620 - 66.010.353

Direzione Amministrativa: Walter Buzzavo

Pubblicità e Marketing
Gruppo Editoriale JCE - Divisione Pubblicità
Via Ferri 6 - 20092 Cinisello Balsamo (MI)
Tel. 02/66025.1

Direttore Publicità: Giuseppe Tiani

Responsabile Marketing: Daniela Morandi

Concessionario eslusivo per Roma, Lazio e centro sud:
UNION MEDIA srl - Via C. Fracassini, 18
00198 Roma - Tel. 06/3215434 (13 linee R.A.)
Telex 630206 UNION I - Telefax 06/3215678

Abbonamenti: Le richieste di informazioni sugli abbonamenti in corso si ricevono per telefono tutti i giorni lavorativi dalle ore 9 alle 12. Tel. 02/66025311- 66025338

I versamenti vanno indirizzati a:
Gruppo Editoriale JCE srl, Via Ferri 6
20092 Cinisello Balsamo (MI), mediante l'emissione di assegno circolare, cartolina vaglia o utilizzando il c.c.p. n. 351205. Per i cambi di indirizzo allegare alla comunicazione l'importo di L. 3.000, anche in francobolli,
e indicare insieme al nuovo anche il vecchio indirizzo.

Radio Elettronica & Computer

SOMMARIO

N° 9 - Novembre 1989

Caricate così i programmi della cassetta allegata

*Riavvolgete il nastro e digitate **LOAD** seguito da **RETURN** sulla tastiera del C64 e **PLAY** sul registratore. Verrà caricato il programma di presentazione con il menù dei programmi. Digitate **RUN** seguito dalla pressione del tasto **RETURN**. Terminata la presentazione, per caricare uno qualsiasi dei programmi è sufficiente digitare: **LOAD "NOME PROGRAMMA"** seguito dalla pressione del tasto **RETURN**.*

C1-Text

Iniziamo con un programma per Amiga serio, anzi professionale. Si tratta di una novità in tutti i sensi: un vero word processor di notevole valore realizzato in Italia. La ditta che coraggiosamente ha deciso di affrontare questo impegnativo compito è la Cloanto di Udine. Accanto alla inevitabile soddisfazione di vedere che anche in Italia è possibile realizzare software di qualità per Amiga si deve tenere presente che solitamente i word processor sono realizzati per essere utilizzati da utenti di lingua inglese, pertanto non possono essere sfruttati nel modo migliore da chi desidera scrivere in italiano piuttosto che in francese o in tedesco.

Naturalmente le difficoltà non consistono nei diversi messaggi che compaiono nel programma: ormai chiunque abbia un po' di dimestichezza con i word processor o semplicemente con gli editor di testo sa benissimo che cosa significa Save, Load, Cut, Copy e così via. Chi anche non conosce questi termini non dovrebbe avere molta difficoltà a impararne il significato in breve tempo. Il motivo per cui è veramente utile che un word processor sia creato appositamente per l'uso con la lingua italiana consiste nella possibilità di utilizzare efficacemente il correttore di errori (solitamente lo si chiama spelling checker, tuttavia data la particolare circostanza è meglio utilizzare il termine nostrano).

C1-Text è dotato di un'ottima gestione degli errori che non solo provvede a segnalare la presenza di parole non contenute nel suo vocabolario, ma addirittura corregge automaticamente gli accenti (se commettete quell'errore che fa inorridire le maestre elementari, ovvero scrivete -pò- con l'accento, nessuno se ne accorgerà perché C1-Text provvederà automaticamente a sostituire la -ò- accentata con la -o- più l'apostrofo). Non possiamo entrare in ulteriori dettagli su C1-Text per non privare i giochi di cui dobbiamo parlare del necessario spazio, tuttavia vi possiamo dire che C1-Text è un ottimo prodotto: certamente non raggiunge il livello di sofisticazione e completezza di alcuni prodotti di oltreoceano come Excellence! oppure WordPerfect, d'altra parte pensiamo che la maggior parte degli utenti possa trovare questo software della Cloanto l'ideale per le proprie necessità poiché si tratta di un programma sufficientemente completo e soprattutto è particolarmente efficace nella nostra lingua. Abbandoniamo frettolosamente i programmi "seri" per passare alla lunga carrellata di videogiochi che ci attende. Premettiamo che tutti i giochi recensiti in questa rubrica sono disponibili sia per C64, sia per Amiga. Solitamente le due versioni dello stesso gioco sono molto simili, tranne eventualmente sia per la grafica e sia per il sonoro che possono sfruttare in modo più efficace l'hardware più sofisticato di Amiga.

Gunschip. Al comando del potente AH-64A

Gunship

Iniziamo con uno stupendo simulatore di elicottero: Gunship. Si tratta di un prodotto Microprose, e certamente il nome di questa software house è una garanzia per quanto riguarda i prodotti di simulazione. In realtà Gunship non è un programma nuovissimo, anzi la versione per C64 risale a un paio d'anni fa, tuttavia la novità, che a nostro avviso è molto importante, consiste nel cambiamento di politica della Microprose. Fino a poco tempo fa questa blasonata software house si rifiutava di esportare i propri prodotti in Italia adducendo come spiegazione la diffusione della pirateria software nel nostro paese. Adesso invece non solo la Leader Distribuzione è stata incaricata della vendita di Gunship nel nostro paese, ma ha anche provveduto a tradurre il manuale di istruzioni in italiano. Poiché Gunship è un programma molto complesso il manuale è essenziale, pertanto la traduzione del medesimo è sicuramente un fatto molto positi-

vo. La confezione di Gunship è molto ricca: oltre al dischetto con il software e ai manuali di istruzione (c'è anche l'originale inglese in cui è possibile trovare oltre alle istruzioni i dati tecnici relativi ad alcune apparecchiature militari sia statunitensi, sia sovietiche) troviamo un'utilissima mascherina da applicare sopra alla tastiera di Amiga.
Su questa mascherina si trovano riportati tutti i comandi dell'elicottero in corrispondenza del tasto che li aziona. Poiché tali comandi sono davvero numerosi la mascherina è un accessorio davvero utile per guidare in modo rapido ed efficace il mostruoso elicottero da guerra. Gunship non è solo un simulatore di elicottero, bensì è un simulatore di azioni di guerra. Attenzione: questo non significa che Gunship sia il solito giochino in cui si deve sparare all'impazzata per uccidere più nemici possibili. Al contrario Gunship è un completo simulatore di elicottero, dotato di tutti i comandi tipici degli elicotteri. La sola simulazione dell'elicottero, con tutte le sue caratteristiche dovrebbe bastare per rendere Gunship un ottimo programma. In più Gunship vi permette di controllare un numero davvero considerevole di armi, poiché Gunship è una fedelissima simulazione di Ah-64A, uno dei più potenti elicotteri da guerra in dotazione all'esercito statunitense.
All'inizio di ogni partita con Gunship vi viene assegnata una missione, completa di tutte le indicazioni necessarie per affrontarla.
Sono possibili diversi livelli di difficoltà che possono soddisfare qualsiasi utente, dal novizio al più esperto. A rischio di sembrare banali dopo quanto vi abbiamo detto di Gunship, dobbiamo aggiungere che si tratta di un programma molto giocabile, ed è dotato di una grafica molto valida, sia nella versione per C64, sia nella versione per Amiga.
Dulcis in fundo il manuale di istruzioni, oltre a essere tradotto in italiano, è anche molto completo: non solo spiega tutti i comandi necessari per guidare l'elicottero e per controllare le armi, ma raccoglie una serie di consigli utilissimi per imparare a controllare nel modo migliore il pesante apparecchio (non è facile guidare un elicottero, sapete!).

Licence To Kill. James Bond in azione, di certo non si tratta del migliore Sean Connery

I nostri più vivi complimenti agli ideatori e ai programmatori di Gunship!

Licence to kill

Dalla guerra sul campo di battaglia alla guerra fredda, ovvero da Gunship a Licence to kill, licenza di uccidere. Avrete tutti capito che stiamo parlando di James Bond, il mitico agente 007 al servizio di Sua Maestà, la regina del Regno Unito. Sull'onda del nuovo film "Licenza di uccidere" l'omonimo videogioco ci porta James Bond sullo schermo del nostro computer. Il sottotitolo del film è molto drammatico e suona pressapoco così:"È solito prendere ciò che vuole... questa volta vuole vendetta".
La drammaticità del programma finisce con il sottotitolo, infatti Licence to kill è un videogioco vistosamente datato, non tanto come soggetto (il film è uscito recentemente), piuttosto come dinamica. Lo scopo del gioco consiste nel superare un certo numero di "scene", ciascuna della quali è suddivisa in diverse parti. In ogni scena il nostro eroe controlla un diverso veicolo e deve riuscire a raggiungere ed eliminare il suo nemico, il contrabbandiere di droga Sanchez. Se questo programma non si riferisse a James Bond probabilmente non sarebbe assolutamente degno di nota, tuttavia il potere del cinema può anche portare a sopravvalutare un videogioco. Non fatevi tentare dal nome del programma e dalla bella presentazione poiché, tanto per parafrasare il titolo di un altro famoso film, "sotto la copertina niente".

Kick Off

Vista l'incombenza dei tanto attesi campionati mondiali di calcio, è lecito aspet-

tarsi un certo proliferare dei giochi basati su questo sport.
Questo mese vi parliamo di Kick Off, la più recente nella lunga schiera di simulazioni calcistiche che ormai imperversano in ogni videoteca. Forse per il fatto che il calcio è lo sport più popolare in Italia, forse per la particolare predisposizione a essere giocato in due (e non singolarmente contro il computer), il pallone da joystick rimane uno dei punti fissi del mercato dei videogame.
Versione dopo versione, comunque, assistiamo sempre a un affinamento delle qualità tecniche che determinano il sempre crescente successo di questo genere di giochi. Anche il gioco in questione, una produzione Anco Software, non è esente da alcune innovazioni o miglioramenti che ne rendono elevata la giocabilità. Per

Kick Off. In attesa dei mondiali allenatevi con Kick Off

rendervene conto riassumiamo qui di seguito le principali caratteristiche: scrolling in ogni direzione del campo (e non solo in senso longitudinale come accade spesso in altri giochi simili), scanner che mostra la posizione di tutti i giocatori (nella versione per Amiga), possibilità di provare schemi di gioco senza la presenza di giocatori avversari, cinque livelli di abilità indipendenti (per esempio una squadra del campionato interregionale può giocare contro una nazionale), molteplicità di opzioni, per esempio la possibilità di scelta della disposizione in campo delle formazioni, torneo di otto squadre (con scelta di quelle da manovrare con il joystick o comandate dal computer), possibilità di salvare un torneo in corso, movimenti dei giocatori immediati e di facili comandi, nove tipi di calci d'angolo, falli puniti con ammonizioni ed espulsioni, fattore campo e vento, tempi supplementari, 12 arbitri con diversi livelli di severità... e si potrebbe continuare!
Lo slogan del gioco, così come appare sulla confezione, è "giocarlo è facile, conoscerlo a fondo richiederà tempo, molto tempo". In effetti la filosofia di Kick Off è proprio questa: anche il giocatore meno esperto potrà divertirsi aggiustando le modalità di gioco secondo la propria abilità; il veterano invece potrà sfruttare tutte le potenzialità del programma mettendo a frutto l'esperienza accumulata.
A differenza della maggior parte delle altre simulazioni calcistiche per controllare la palla e correre con essa non è sufficiente porvi il giocatore sopra e manovrare il joystick nella direzione desiderata; come avviene nella realtà il giocatore deve procedere a piccoli tocchi e nei cambi di direzione è necessario che prima aggiri la palla. In sintesi i giocatori non posseggono il piede a calamita per cui la palla li segue qualsiasi movimento essi compiano. Anche se prima del fatidico giugno 90 ne vedremo molti altri simili (e probabilmente anche dopo), a questo gioco spetta sicuramente un posto di riguardo fra quelli del suo genere soprattutto per il fatto di presentare una sintesi di tutti i migliori aspetti di precedenti versioni di calcio al computer. Inoltre è supportato da una buona grafica e, particolare non irrilevante, da scritte in quattro lingue fra cui l'italiano. Certamente, però, la traduzione poteva essere un po' più curata.

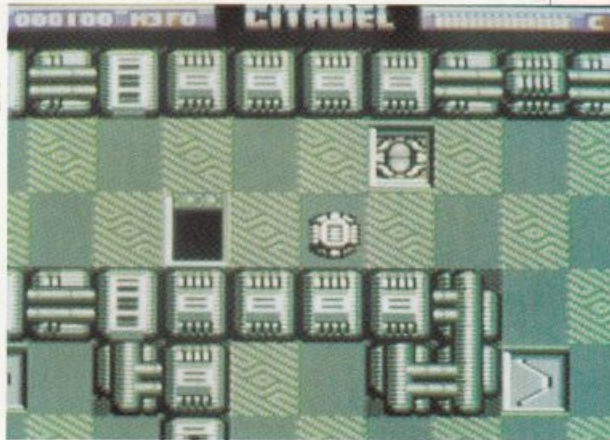
Red Heat

Veniamo ora a una rassegna di giochi d'azione partendo dal più atteso, Red Heat, un programma della Ocean che riprende il filone dei giochi ispirati ai film di Arnold Schwarzenegger. Proprio nel numero scorso avevamo presentato Commando ed eccoci già a riparlare dell'austriaco tutto muscoli e della sua interpretazione di "Danko", recentemente proiettato sugli schermi italiani. Il film, come il gioco, tratta delle vicende di Ivan Danko, capo della sezione omicidi della polizia moscovita, in missione a Chicago per rintracciare Viktor Rostavili, boss di una banda internazionale di trafficanti di droga. L'azione

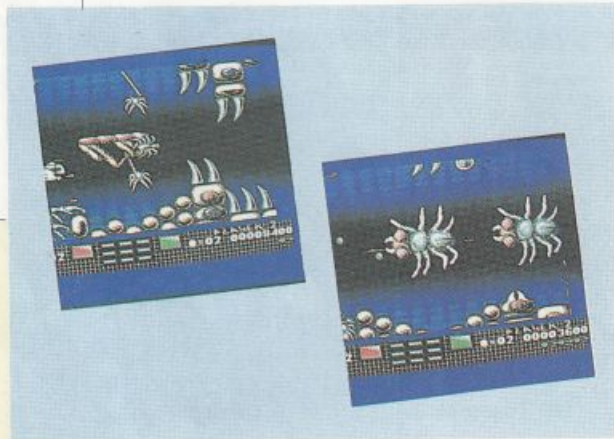
consiste in spietati corpo a corpo contro squadre di criminali che cercano di coprire la fuga del loro capo, fino alla resa dei conti con Rostavili in persona.
Il gioco si svolge su quattro livelli di difficoltà crescente nei quali appaiono alcune fasi che elargiscono punti da aggiungere a quelli che ci si è procurati avanzando e vincendo gli aggressori.
Determinati punteggi permettono il recupero dell'energia dispersa durante la lotta e l'aumento della potenza di tiro nel caso durante il cammino si venga in possesso di una pistola. Altri oggetti potranno essere raccolti ma non tutti saranno d'aiuto per raggiungere il trafficante ricercato. Red Heat si distingue più per la grafica e il realismo dei movimenti che per l'originalità del tema. La giocabilità risente abbastanza della ripetitività dell'azione all'interno dei singoli livelli. Singolare il fatto che sia Danko sia i suoi nemici appaiano sempre di mezzo busto, cosicché nei corpo a corpo è possibile usare solo gli arti superiori ed eventualmente la testa.

Citadel

Passiamo a quella che probabilmente è la novità più interessante di questo mese; Citadel, produzione Electric Dreams Software, è un arcade di ottima fattura che si svolge in una avvenieristica città spaziale multipiano. Al livello più profondo di essa sono ancora nascosti i segreti di un'antica civiltà che si presumeva estinta; a bordo di una particolare sonda, il Monitor, vi avventurerete nei meandri degli otto piani collegati da ascensori, ognuno dei quali nasconde imprevedibili sistemi di difesa.
I percorsi per passare da un ascensore all'altro sono obbligati e quindi non potrete evitare di incorrere in torrette laser che spuntano dal sottosuolo, barriere di energia che dissolvono qualsiasi cosa le attraversi, automi omicidi che danno la caccia non appena percepiscano il minimo movimento. Questi ostacoli possono però diventare utili nel caso li abbiate catturati; per esempio le barriere che difendevano le torrette possono essere sfruttate come scudi, così come i droidi che una volta colpiti vi seguono proteggendovi dal fuoco nemico. Il Monitor dispone di un braccio meccanico che può prendere qualsiasi cosa incontri sulla strada, dai rifornimenti di munizioni a quelli di energia. Inoltre esso può manovrare l'interruttore che attiva e disattiva la barriera di energia più vicina. Il braccio meccanico è utilizzabile solo se si possiede energia sufficiente a farlo funzionare. Il punteggio si incrementa con la distruzione delle difese ed il completamento dei vari piani; ogni diecimila punti si ottiene un Monitor in aggiunta ai due di dotazione base. Citadel associa il dinamismo tipico dell'arcade alla riflessione e all'astuzia necessari per i giochi strategici; infatti, poiché le difese non agiscono se si è fermi e non si spara, è possibile compiere delle soste per riflettere sulla prossima mossa e organizzare le sortite successive. La grafica è molto buona come del resto gli effetti sonori che accompagnano l'azione.

Citadel. Azione e strategia nell'avvincente avventura di Citadel

Phobia

La paura è parte inalienabile della mente di ogni umano e questo Lord Phobos, signore del male, lo sa molto bene. Egli ha rapito la figlia del Presidente Galattico e, confidando sul fatto che una particolare fobia possa scoraggiare anche l'eroe più indomito, ha creato dei mondi particolari che separino il Sole, dove ha la sua base, dalla Terra. Questi mondi, infatti, sono immensi scenari rappresentanti ognuno una determinata fobia, da quella per i ragni a quella per gli scarafaggi, da quella per il fuoco a quella per la morte e altre ancora. Chi sarà l'ultracoraggioso che andrà in soccorso della fanciulla rapita? Ovviamente colui che si cimenterà in Phobia, l'ultima produzione della Image Works. Si tratta di attraversare nove dei pianeti fobici ognu-

no dei quali racchiude al suo interno un elemento molto raro che assicura protezione totale dal calore del sole. Superando indenni tutti quanti i mondi si accede al Sole, a meno che non ci si imbatta in una delle barriere di luce collocate fra i pianeti dalle temute Truppe solari. Questi ostacoli possono essere superati solo sacrificando una navicella o adoperando una Capsula spaziale raccolta sul percorso effettuato fino a quel punto. Sia l'esplosione delle Capsule sia quella dei pianeti fobici sprigionano energia che può essere immagazzinata dalla navicella per migliorare la potenza di fuoco.
Phobia può essere giocato singolarmente o in due persone: nel primo caso sarà possibile sdoppiare la navicella aumentando la potenza distruttiva ma anche l'esposizione agli attacchi nemici; nel secondo si

Phobia. Una immagine dalla confezione

potrà utilizzare il Sistema Offensivo Speciale di Scambio Energetico per cui, sparando sul compagno, si otterrà l'effetto di caricare la sua navicella rendendone migliore il potere distruttivo.
Questo programma è un esempio di come sia possibile rinnovare la giocabilità di un genere già ipersfruttato (quello delle incursioni spaziali in territori ostili) apportando alcune modifiche originali e ben congegnate.
Per esempio i raid sui pianeti non sono fine a se stessi ma fanno parte di un unico filo conduttore rappresentato dalla ricerca degli elementi di protezione; anche il reciproco caricamento delle navicelle alleate è un'idea molto ben azzeccata. Gli scenari in cui si svolge l'azione sono inconsueti e di notevole resa grafica.

Forgotten worlds

Un tema abbastanza simile a quello del gioco precedente è trattato in Forgotten worlds, prodotto dalla Capcom. Anche in questo caso si è protagonisti di una incalzante avventura in territori ostili, mondi che degli dei maligni, comandati dal terribile imperatore Bios, hanno ridotto a un ammasso di rovine distruggendo ogni civiltà che vi abitava. Unica possibilità di fermare l'avanzata distruttiva di questo olimpo del male è quella di risalire uno per uno i mondi sottomessi fino ad arrivare sopra le nuvole, dove Bios stesso è pronto a combattere. Tutti e quattro i livelli si concludono con l'eliminazione di uno degli dei che presidiano le rovine: ognuno di essi possiede un punto vulnerabile, chi la bocca, chi il cuore, chi le spalle; offendendo le altre parti dei loro mostruosi corpi si sortisce solo l'effetto negativo di aumentarne la forza e la furia combattente.
All'interno dei mondi non si è dotati di alcun mezzo di locomozione se non di un sistema di retrorazzi che permettono di librarsi facilmente in aria. Le armi disponibili sono di diverso tipo (missili, bombe, cannoni, razzi, lanciafiamme...) e possono essere comperate al termine di ogni livello. È possibile anche munirsi di corazza e di un dispositivo che aumenta il potere distruttivo delle armi.
Forgotten worlds non è proprio una pietra miliare nel campo dei videogame ma in ogni caso si allinea in quella vasta schiera di programmi che, anche se non a lungo, possono offrire vari spunti di divertimento. Il semplice fatto che possa essere giocato in contemporanea da due giocatori offre l'opportunità di avvincenti sfide. Forse qualche miglioramento poteva essere apportato per quel che riguarda la grafica, con particolare riferimento alla versione per C64.

Anteprime

•Ocean

"Prossimamente su questi schermi": quante volte abbiamo sentito questa frase in un cinema per annunciare l'uscita di un nuovo film! Per "The untouchables"(Gli intoccabili) gli schermi in questione non sono solamente quelli dei cinema o dellaTv, ma anche quelli dei nostri computer.

La Ocean Software ha annunciato con un certo clamore l'uscita di "The untouchables" per tutti i computer più diffusi.
Poiché la Ocean ci ha abituato a prodotti di qualità sin dai tempi in cui il C64 era ancora in fasce e dell'Amiga non esisteva neppure il progetto, pensiamo che quello che loro chiamano "uno dei maggiori e più ambiziosi progetti mai intrapresi nella storia dei videogiochi" sia un programma degno di molta attenzione.
Si parla di un programma suddiviso in cinque parti, in cui voi controllate la squadra scelta di Eliot Ness. La storia su cui si basa il film e quindi il programma è la battaglia condotta da Eliot Ness negli anni 20 contro la mafia e in particolare Al Capone. Sean Connery ha vinto un Oscar per la sua interpretazione nel film: voi sarete capaci di imitarlo?

•U.S. Gold

Restiamo nel mondo dello spettacolo passando dal cinema alla musica pop, in particolare Michael Jackson. Se la Ocean Software presenta in pompa magna il suo The untouchables certamente la U.S.Gold, altra software house degna della massima considerazione, non è da meno ed esordisce nel seguente modo: "Nel 1969 Neil Amstrong fu il primo uomo a camminare sulla Luna, vent'anni dopo Michael Jackson camminerà sulla Luna attraverso gli schermi dei computer".
In questo modo la U.S.Gold ci annuncia l'uscita in questi giorni di Moonwalker per tutti gli home computer più diffusi. Non sono disponibili altre notizie: non appena verremo a conoscenza di qualche particolare in più non mancheremo di informarvi.

•Palace Software

La Palace Software ha annunciato l'uscita di The Dungeon of Drax, la versione per Amiga di Barbarian II. La differenza più apprezzabile rispetto alla versione precedente consiste nella modifica delle routine di controllo della grafica e del suono. In particolare si è fatto abbondante uso di suoni digitalizzati sfruttando al massimo i quattro canali audio di Amiga. Naturalmente Barbarian II è disponibile anche per Commodore 64, sia su cassetta sia su disco.

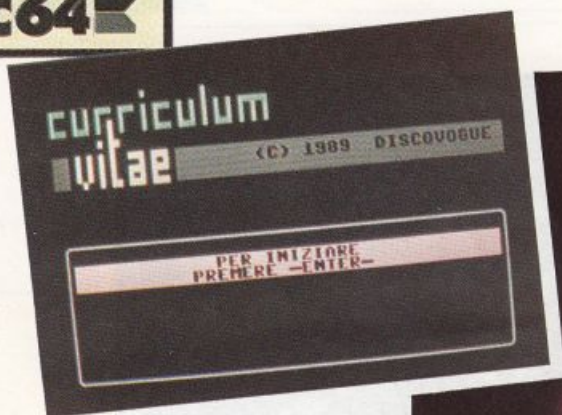
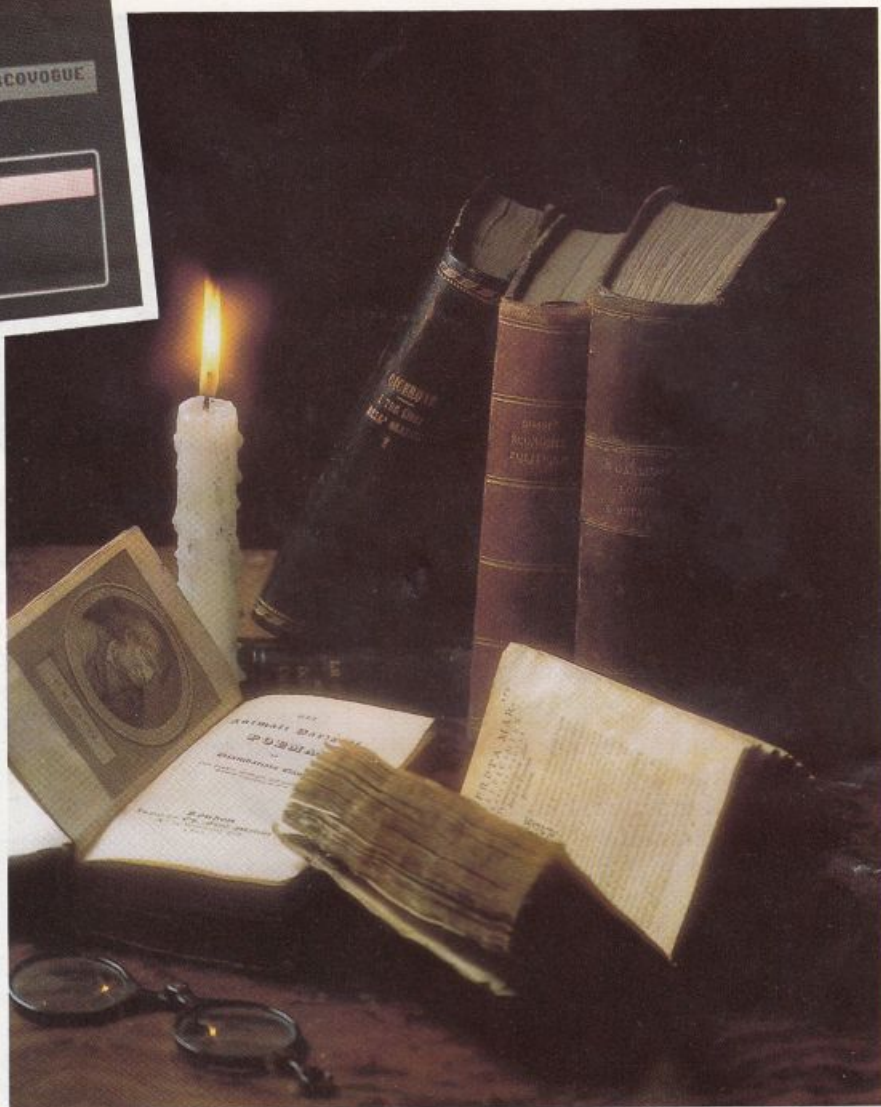
Forgotten Worlds. 4 livelli di incessanti insidie nei "mondi dimenticati"

Ancora la Palace Software ci informa dell'uscita di Castle Warrior per Amiga. Castle Warrior è un arcade dalla grafica molto sofisticata che può essere implementata in modo efficace solo su macchine da 16 bit. Manca quindi la versione per C64, mentre è disponibile la versione per Atari St, che come è noto utilizza lo stesso microprocessore di Amiga.

•Infocom

Amanti dei racconti "cappa e spada" che possedete Amiga fate attenzione: la Infocom ha presentato Arthur, the quest for Excalibur. Si tratta di un programma grafico interattivo che vi conferisce il nobile ruolo di Arthur (o Artù se preferite), un giovane cavaliere alla ricerca di Excalibur, la spada magica che può assegnare al suo proprietario il trono di Inghilterra.
Inutile dire che la spada è stata rubata da Lot, un re malvagio, e che se voi la volete conquistare dovrete affrontare una serie di imprese che verificheranno le vostre capacità di cavaliere.
Non manca comunque l'aiuto del mago Merlino, i cui magici poteri vi permetteranno di affrontare le numerose avversità che vi ostacoleranno la strada.

Gianni Arioli
Massimiliano Del Rio

Tutto il software citato in questa rubrica è distribuito in Italia da: Leader distribuzione Srl - Via Mazzini, 15 - 21020 Casciago (Va) - Tel. 0332/212255.

Grazie al software di questo mese e all'aiuto della vostra stampante Commodore o compatibile, troverete facilissimo mettere a punto un'ottima lettera sul vostro Curriculum Vitae per presentarvi al meglio a un qualsiasi datore di lavoro potenziale

Il C64 insegna!

Il programma Curriculum vitae permette di definire un adatto formulario per la stesura, tramite stampante, di qualsiasi tipo di curriculum vitae, cioè di quel documento che ogni persona in cerca di lavoro (in particolare impiegatizio o manageriale) è solitamente tenuta a fare quando cerca un'occupazione presso un nuovo soggetto datore di lavoro: questo può essere per esempio una ditta, un ente della pubblica amministrazione, un commerciante o un professionista. Curriculum vitae è una definizione latina che al giorno d'oggi può essere tradotta come elenco delle tappe fondamentali della vita, ovviamente riferite nella fattispecie alla formazione scolastica e post-scolastica di una persona.

Il documento curriculum vitae è praticamente il biglietto da visita che si pre-

senta all'addetto alla selezione del personale, e per questo dev'essere redatto a regola d'arte, composto con chiarezza, senza dimenticare alcuna precisazione utile a qualificare ulteriormente chi lo stila: in fondo l'impressione che ne ricava chi lo legge sarà più o meno positiva proprio in base all'ordine d'esposizione delle qualifiche e delle referenze, fermo restando che le doti e le credenzialità personali saranno sempre il parametro di riferimento per ogni considerazione finale.

In tempi in cui oltre alla sostanza si tende a dare notevole importanza a tutto ciò che "appare dunque è", e cioè anche alla forma, un curriculum vitae ottimamente preparato aiuterà chiunque, nelle selezioni o nei concorsi, a essere preferito rispetto a ipotetici "concorrenti" dotati di pari credenziali esposte però in modo disordinato, superficiale o di difficile lettura. Il documento che si può ottenere è direttamente utilizzabile da chi lo deve sottoscrivere e presentare: potrà essere lasciato a chi ci fa l'audizione, senza scordare che anche quando non richiesto sarà sempre gradito e utile per indicare referenze o recapiti, o altri dati personali.

Curriculum vitae è un programma adatto a comporre fogli veramente completi, ed essendo autostrutturato evita ai più distratti dimenticanze che possono poi diventare causa di cestinamento: quante volte capita di dilungarsi in descrizioni inutili e poi ci si scorda di mettere il proprio indirizzo o la data di nascita, oppure specifiche indispensabili come posizione di servizio militare o nomi di affidabili garanti (senza raccomandazioni non si va mai tanto avanti).

Una volta ottenuta la stampa del curriculum è sufficiente sottoscriverla con firma leggibile, per renderla presentabile, in quanto è a tutti gli effetti una normalissima scrittura privata non avente alcun valore giuridico (sempre che se ne faccia uso in buona fede!).

Curriculum vitae è insomma un utile strumento di lavoro ideale non solo per chiunque voglia fare buona impressione a un ipotetico datore di lavoro in cerca di personale, ma anche proprio per la controparte, ovvero per chi, dovendo selezionare il candidato giusto per un determinato posto, decida di razionalizzare tutta la fase di audizione e scrematura magari compilando via computer tutti i curriculum, che saranno dunque di omogeneo aspetto e dunque più agevolmente comparabili. All'utente viene semplicemente chiesto di specificare, tramite tastiera, tutti i dati e le credenziali utili a formare un buon pacchetto, tra cui, oltre alle generalità personali e al curriculum scolastico, anche corsi di perfezionamento, lingue straniere conosciute, esperienze di lavoro e referenze supplementari. Su questi input il computer lavora velocemente, definendo volta per volta un formulario ad hoc e trasmettendo alla stampante (quella di riferimento è la Commodore Mps 1200) il messaggio software per la produzione del documento cartaceo, che avviene su normali fogli formato A4 (questi diventano sempre, per un curriculum di media lunghezza, almeno due).

Tramite un unico paginone grafico multicolor si può controllare con facilità lo svolgimento di tutte le routine di elaborazione, con monitoraggio effettuato da un multidisplay che permette una lettura istantanea e soprattutto facilità di interpretazione: il tutto senza possibilità di errori.

La carica sul Commodore 64 si effettua con i consueti comandi di loading, ovvero LOAD"" oppure LOAD"FRST2" (visto che il file ha come nome FRST2), o anche con il pratico metodo di pressione contemporanea dei tasti Commodore e Run/Stop, che causa tra l'altro l'autostart immediato a fine carica, evitando la necessità del Run attivatore. In memoria Ram il programma occupa uno spazio di poco più di 5 Kb, che aumentano di circa 0,5 Kb a lancio avvenuto.

Funzionamento e uso

Terminata l'operazione di caricamento, trascorsi pochi secondi da quando il programma viene fatto partire (tramite un RUN di lancio) compare la pagina-

monitor a fondo e bordo di colore nero, che rimane invariata nella sua parte grafica e strutturale durante tutte le fasi di esecuzione delle routine.

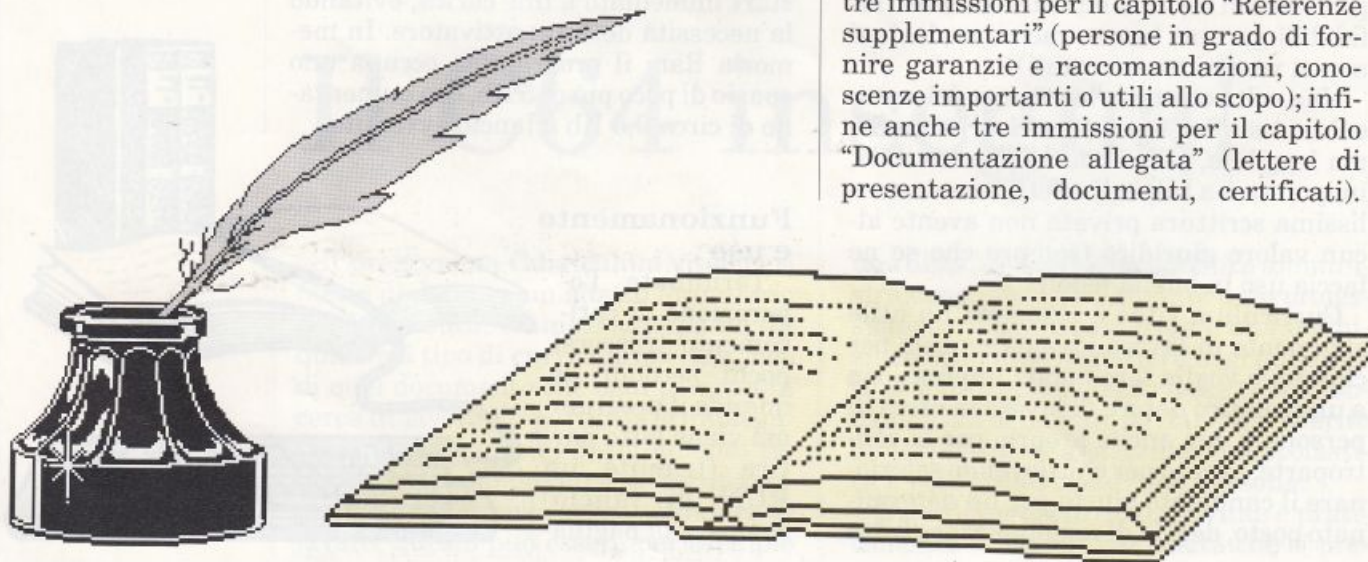
Nella zona superiore dello schermo sono presenti la scritta "Curriculum vitae" (nei colori verde e giallo) e il data set di copyright (in colore grigio reverse).

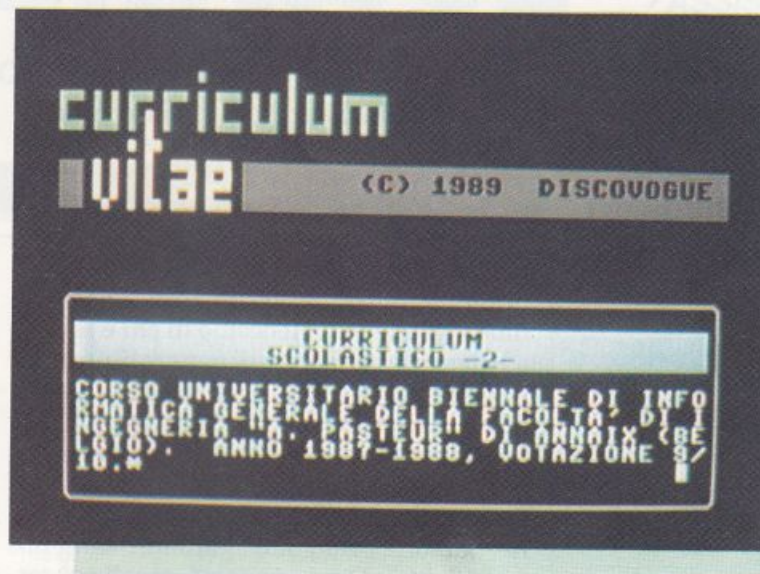
Nella parte inferiore dello schermo,che è evidenziato da una cornice bianca, trova sistemazione il multidisplay per la scrittura dei messaggi-guida, per la visualizzazione degli input e per la proiezione della videata finale di stampa (in standby).

Anche se non esplicitamente precisato su video, si intende che l'uso del programma Curriculum vitae non comporta, comunque e da chiunque venga usato, alcuna responsabilità per chi lo ha realizzato, prodotto e pubblicato: occorrerà pertanto controllare sempre attentamente ogni operazione che porta poi alla stampa del documento. Il messaggio iniziale in colore rosa reverse.

Per iniziare premere Enter

Avverte che si può iniziare attivamente solo premendo il tasto Return sul Commodore 64: in caso contrario il computer rimane in attesa. Si inizia con la richiesta dei dati di inevitabile citazione: nell'ordine occorre specificare luogo e data di stesura del curriculum, quindi le generalità del destinatario (che esaminerà il documento) e dello scrivente (che lo presenterà) elencando per entrambi cognome, nome e indirizzo completo. Di seguito occorre precisare solo informazioni relative allo scrivente: luogo e data di nascita innanzitutto, poi stato civile (per esempio coniugato), descrizione della persona (aspetto, presenza, carattere, abitudini, predisposizioni), posizione circa gli obblighi di leva per gli uomini (per esempio militesente), e infine altre notizie (disponibilità a viaggiare o a orari di lavoro particolari, eventuale auto posseduta se richiesta, recapiti telefonici).

Questi input verranno poi stampati sul documento in un'apposita sezione denominata "Informazioni preliminari". Da questo punto in avanti gli input sono di facoltativa immissione, perché dipendenti dal curriculum di ciascuna persona: il programma prevede comunque un plafond massimo di ben: quattro immissioni per il capitolo "Curriculum Scolastico" (grado di istruzione raggiunto, votazioni, diplomi e lauree conseguiti e corsi integrativi frequentati); due immissioni per il capitolo "Corsi post scuola perfezionamento " (stage applicativi, training, esperienze teorico-pratiche); tre immissioni per il capitolo "Lingue straniere conosciute" (livello di apprendimento, parlato e/o scritto, integrale o solo commerciale); quattro immissioni per il capitolo "Precedenti esperienze di lavoro" (corsi a contratto di formazione, periodi di apprendistato o praticantato, assunzioni temporanee, stagionali e permanenti); tre immissioni per il capitolo "Referenze supplementari" (persone in grado di fornire garanzie e raccomandazioni, conoscenze importanti o utili allo scopo); infine anche tre immissioni per il capitolo "Documentazione allegata" (lettere di presentazione, documenti, certificati).

Durante le fasi di immissione dei dati un'apposita maschera di input fornisce di volta in volta un messaggio-guida affinché l'utente dia sempre risposte adeguate: in verde chiaro reverse viene stampato l'argomento della richiesta, e più sotto un marker dello stesso colore delimita un campo in cui viene poi visualizzato l'inserimento, la cui ampiezza è variabile (si va dai cinque caratteri per l'indicazione di un codice di avviamento postale a un massimo di ben 178 caratteri per una specifica descrittiva).
Alcuni input prevedono l'uso dei soli tasti numerici (dallo zero al nove), mentre altri anche di quelli alfabetici, grammaticali e semigrafici (punti, virgole, segni e barre molto utili per esempio per scrivere bene gli indirizzi): la visualizzazione su video avviene in colore verde chiaro reverse, da sinistra verso destra (tipo word processor). Un asterisco mobile (cursore) indica la posizione raggiunta dal testo inserito rispetto allo spazio disponibile.
La conferma di ogni inserimento va data con il tasto Return, mentre eventuali correzioni parziali sono sempre possibili, prima della conferma stessa, utilizzando il tasto della freccietta sinistra (sul Commodore 64 è posto di fianco a quello numerico 1): ciò causa l'arretramento di un passo del cursore e la cancellazione di un carattere. Correzioni totali si possono disporre premendo invece il tasto Inst/Del: in questo caso il testo relativo all'input al momento su video viene totalmente cancellato per essere riscritto daccapo.
Premendo subito e solo il tasto Return, senza cioè digitare caratteri di input, si passa oltre: l'immissione sarà allora considerata come inesistente, nulla, e non comparirà sul curriculum (si procede così quando per esempio il curriculum scolastico consiste in meno delle quattro specifiche possibili previste).

Un elegante automatismo permette di non stampare capitoli privi di almeno uno degli input a disposizione: se per esempio non si conoscono lingue straniere e non si utilizza nessuno dei tre input previsti, sul curriculum si passerà dal capitolo "Corsi post scuola di perfezionamento" direttamente al capitolo "Precedenti esperienze di lavoro". Al completamento dell'esecuzione della routine di immissione dei dati viene subito proiettata la videata finale con il messaggio in color rosa reverse.

Eseguzione stampa documento

Che ricorda all'utente di aspettare affinché la stampante collegata al computer provveda alla stesura del curriculum definito. È possibile scegliere i fogli singoli in formato standard A4 (da inserire in sequenza) oppure i moduli continui, anche se ovviamente la prima soluzione è quella da preferire: infatti normalmente i curriculum sono fogli a più facciate o pagine. Per ottenere l'eventuale scrittura in fronte/retro è sufficiente reinserire nella stampante un foglio appena scritto ed espulso dal rullo, avendo ovviamente cura di capovolgerlo senza girarlo.

La lunghezza di una scrittura-tipo, con la spaziatura interlineare prevista (6 millimetri) diventa sempre di almeno due fogli, ovvero uno se si opera in fronte/retro (in quest'ultimo caso se si ottengono più fogli vanno uniti con colla o punti metallici controllando la corretta impaginazione del documento). Normalmente ogni curriculum viene prodotto in due copie: una viene lasciata dalla persona esaminata all'addetto alla selezione, mentre l'altra viene conservata per eventuali controlli. Il programma ne stampa una sola, per cui si può ripetere la procedura di definizione oppure, meglio, ricorrere all'aiuto di una fotocopia-

trice. Sulla carta ogni settore del curriculum è opportunamente evidenziato da un titolino in reverse, e ogni specifica è segnalata da marker (rettangolini sulla sinistra del foglio), mentre il testo è stampato in Nlq (near letter quality) per un'ottima leggibilità. La composizione dei contenuti è automaticamente giustificata ai margini, al fine di conferire all'elaborato un'impaginazione veramente professionale e gradevole all'occhio di chi è tenuto a leggere la sequenza di generalità personali e referenze. Il curriculum assume validità quando viene sottoscritto da chi lo compone, mediante firma leggibile da apporre nell'apposita riga già prevista alla fine del documento (nome e cognome della persona vengono automaticamente specificati). Ovviamente è meglio prendere visione dell'elaborato, prima di sottoscriverlo e di presentarlo a chi lo ha richiesto e ne deve far uso: trattandosi di una normale scrittura privata può essere cambiata (riscritta) senza difficoltà ogni volta che intervenga necessità (per esempio per aggiornamenti o modifiche).

La stampante di riferimento che garantisce i migliori risultati di impaginazione è la Commodore Mps 1200: altre printer possono produrre documenti di diverso lay-out per quanto riguarda grafica, caratteri e composizione, anche se ovviamente il contenuto rimane sostanzialmente invariato. Curriculum vitae ritorna automaticamente alla videata iniziale di attesa non appena termina la routine di stampa dei documenti di volta in volta definiti.

Concludendo, si può affermare che questo programma aiuta molto l'utente, effettuando tutte le procedure più noiose e delicate che richiede la stesura di un curriculum vitae completo e presentabile a persone attente ed esigenti. Un consiglio prezioso per chi si appresta a stilare un documento: soprattutto nelle (numerose) immissioni di carattere descrittivo è bene scrivere ogni notizia utile a una migliore e più approfondita presentazione personale, senza tralasciare alcun dato. Allo stesso tempo occorre però non dilungarsi eccessivamente: bisogna cioè operare componendo i testi con taglio giornalistico, ovvero completo ma conciso, al limite meglio telegrafico che arzigogolato. È di seguito riportato un esempio di curriculum, anche al fine di mostrare come sia facile ottenere via software risultati immediati e completi. Si precisa che ogni riferimento a luoghi, persone e situazioni eventualmente esistenti è puramente casuale. Si tratta di una richiesta di lavoro fatta dal signor Giancarlo Martellani, abitante a Pontone sul Naviglio (Mi), Corso Adriano 52/B, alla Geotecnica S.p.A. con sede in Carpi (Mo), Piazza della Repubblica 47: questa società ha richiesto il curriculum vitae che il signor Martellani ha stilato e sottoscritto a Milano il giorno 23 settembre 1989. L'addetto alla selezione della Geotecnica S.p.a. può così facilmente conoscere che Giancarlo Martellani è nato a Modena il 21 novembre 1962, risulta celibe, ha come caratteristiche personali una bella presenza, l'attitudine ai rapporti interpersonali ed è un tipo decisamente estroverso. Ha già fatto il servizio di leva, possiede un'auto, è disponibile a viaggiare e a lavorare sia in part-time sia di notte, può essere contattato a casa al numero telefonico 059-11.23.47.63.

Ha un buon curriculum scolastico, comprendente un diploma di maturità scientifica con votazione 56/60, un altro diploma universitario di informatica con voto 9/10 e un attestato di frequenza a un corso di programmazione applicata. Conosce l'inglese commerciale e ha qualche nozione di francese e spagnolo.

C'è poi un lungo elenco di esperienze di lavoro già fatte: tecnico riparatore a Modena nel 1984, commerciante fino al 1986, assicuratore a Bologna negli anni 1987 e 1988, e infine rappresentante per il centro-nord Italia della Pirelli fino al gennaio 1989. Il Rag. Mazzoni di Soliera (Mo) e il Dott. Aldrovandi di Milano sono le persone a cui rivolgersi per eventuali garanzie, referenze e raccomandazioni.

Il signor Martellani dichiara di allegare infine copia della dichiarazione 740/88 e del prospetto Enasarco numero 3483.

Daniele Malvasi

• PERMUTE

• RIPARAZIONI

• ASSISTENZA

OFFERTE SPECIALI • VENDITA PER CORRISPONDENZA • SCONTI RISERVATI AI RIVENDITORI

ARTICOLO	PREZZO	ARTICOLO	PREZZO
Amiga 500 con mouse e 3 dischi	739.000 *	Joystick Flashfire con 3 spari manuali	10.000
Commodore 64 New + Registratore + Penna Ottica, ecc.	339.000 *	Joystick Flashfire con 3 spari man + autofire	15.000
Commodore 128 + Registratore + Penna Ottica	350.000 *	Joystick Flashfire con autofire trasparente	19.000
Disk Drive compatibile x 64/128k	250.000 *	Joystick Flashfire con Microswitches B.	29.000
Disk Drive 1541 II	295.000 *	Joystick Albatros microswitches	49.000
Drive esterno x Amiga 500/2000 multi/disconn.	165.000 *	Joystick Flashfire x C16	15.000
Stampante Riteman x C 64/128K	390.000 *	Joyboard	49.000
Stampante per Commodore 64/128k	325.000 *	Joystick x Amstrad con sdoppiatore D.	24.000
Stampante Commodore 1230 x C64 Amiga e PC	490.000 *	Joystick x Amstrad con sdoppiatore D Microsw	32.000
Stampante Epson x Amiga e Pc	395.000 *	Joystick Digitale a sfioramento	45.000
Stampante Star Lc 10 Colore x Amiga e Pc	495.000 *	Joystick WIZ Master	35.000
Monitor Monocromatico con audio (universale)	165.000 *	Joystick Sinclair Plus 2	24.000
Monitor colore Commodore 1802 x C64 e Amiga/m	330.000 *	Paddles Controllers (coppia)	18.000
Monitor colore 1084 C x C64 Amiga, PC	520.000 *	Adattatore x Joystick C16	5.000
Monitor Philips x C64 Amiga PC	460.000 *	Mouse x C64 128K	60.000
TV Monitor Universale	580.000 *	Mouse x Amiga 500	95.000
Registratore Compat. x C64/128k	50.000 *	Mouse Elettronico X C64 128K Amiga	45.000
Registratore Commodore x C64/128k	68.000 *	Penna ottica x C64/128k cassetta o disco	35.000
Registratore 1531 x C16	68.000 *	Geos x C64/128K	49.000
Adattatore Registratore da C16 e C64	25.000	Regolatore Elettronico di testine reg.	19.000
Adattatore Duplicatore x registratore C16	18.000	Fast Disk (velocizzatore + copiatore C64)	45.000
Duplicatore x Registratore C64	18.000	Final Turbo IV New	79.000
Modulatore x Amiga 500	45.000 *	Power Cartridge	120.000
Espansione di Memoria 512K Amiga 500	250.000 *	Isepic Tape (sprotettore e + copiatore C64)	65.000
Videodigitalizzatore Amiga 500	200.000	Reflex Backup C64	49.000
Videon (digitalizzatore Amiga)	390.000	Turbo Dos	90.000
Video Gen Lock (miscelatore di immagini Amiga)	450.000 *	Speed Dos 1541 II	90.000
Interfaccia Musicale "Midi" x Amiga 500	120.000	Digitalizzatore x C64	60.000
Copricomputer Plexiglas Amiga 500	19.500	CartucciaCpm x C64	100.000
Copricomputer Plexiglas 128K	18.000	Rom x Stampante Mps 803	48.000
Copricomputer Plexiglas C64 New	16.000	Portacassette "Posso" 15 Pz.	15.000
Copricomputer Plexiglas C64 C16 Vic 20	14.000	Portadischetti 5" 1/4 "Posso"	38.000
Modulatore x Vic 20	35.000 *	Portadischetti 3" 1/2 "Posso"	38.000
Cavo Start x Amiga 500 Tv e Monitor	27.000 *	Portadischetti 3" 1/2 (cont 10 Pz.)	4.500
Cavo Start C64/128k/C16/Vic 20	19.000 *	Portadischetti 3" 1/2 (cont 25 Pz.)	18.000
Cavo Monitor C64/128K/C16	25.000	Portadischetti 3" 1/2 (cont 40 Pz.)	24.000
Cavo Monitor per Amiga 500	40.000	Portadischetti 3" 1/2 (cont 80 Pz.	28.000
Cavo Monitor Colore x PC	59.000	Portadischetti 5" 1/4 (cont 10 Pz.)	4.500
Cavo Monitor videocomposto x PC	12.000	Portadischetti 5" 1/4 (cont 50 Pz.)	24.000
Cavo Monitor videocomposto x C64	14.000	Portadischetti 5" 1/4 (cont 100 Pz.)	28.000
Cavo TV/Computer mt. 1,5	6.500	Portacassette a valigetta 16 Posti	14.000
Cavo TV/Computer mt. 3	12.000	Portacassette a valigetta 36 Posti	18.000
Filtro antidisturbo x Computer	19.000	Portacassette Componibili (multibox)	3.500
Cavo 40/80 colonne x 128k anche Skart	25.000	Dischetti 3" 1/2 Df Dd Conf. 50 Pz. (1 Mega)	CAD 2.000
Cavo Seriale x Drive e Stampante	16.000	Dischetti 3" 1/2 Df Dd Conf. 20 Pz. (1 Mega)	CAD 2.500
Cavo Centronics	25.000	Dischetti 3" 1/2 Df Dd Sciolti (1 Mega)	CAD 3.000
Alimentatore Universale	28.000	Dischetti 5" 1/4 Df Dd Conf. 10 Pz.	CAD 1.000
Caricatore Batterie e Alimentatore	39.000	Dischetti 5" 1/4 Df Dd xc PC	CAD 1.500
Alimentatore x Atari 2600	35.000 *	Dischetti 5" 1/4 Df Dd 3M	CAD 6.000
Alimentatore x Commodore C16	35.000 *	Tagliadischetti in Acciaio	16.000
Alimentatore Commodore 64	45.000 *	Tagliadischetti in Plastica	12.000
Cassette Vergini x Computer da C 10 e C 120	da 1.000	Kit Puliscitestine x Drive	15.000
Videoregistratori e cassette video	Telefonare	Giochi Disco x C 64/128/Amiga/PC/Msx ecc. da L.	10.000
Kit Pulisci testine x Videoregistratore	19.000	Giochi Cassetta x C64/MSX/C16/Atari ecc. da L.	9.000
		Disponibili integrati e ricambi per computer	Telefonare

★ I PREZZI INDICATI CON ASTERISCO SI INTENDONO + IVA 19%

ELECTRONICS PERFORMANCE

Via S. Fruttuoso, 16/A - Monza (S. Fruttuoso) - Tel 039/744164

ACCESSORI

Il futuro in mano

Sui numeri scorsi abbiamo parlato di vari accessori per Commodore 64 prodotti da Gp Elettronica. Fra questi ha suscitato molto interesse il joystick a sensori, cioè un joystick in cui la consueta leva di controllo è sostituita da quattro sensori per la direzione e due per la funzione Fire

Joystick a sensori! Basta un tocco delicato delle vostre mani per gestire qualsiasi tipo di programma che sia studiato per il joystick tradizionale. Grazie a queste caratteristiche, la bella confezione in cui si presenta (in cui è incluso un simpaticissimo astuccio per contenere lo strumento) e il suo aspetto accattivante non sono certamente gli unici pregi di questo rivoluzionario strumento. Abbiamo già sottolineato, nelle precedenti occasioni, la superiore velocità e precisione: l'assenza di organi meccanici cancella ogni tipo di ritardo dovuto allo spostamento dei contatti. Anche la sensazione tattile è molto piacevole, in

quanto ci si accorge di avere un controllo molto sensibile sullo strumento. Non dimentichiamo inoltre che l'assenza di leve e contatti rende Flash Fire a sensori indistruttibile a meno di trattarlo in modo assolutamente inusuale. Abbiamo anche segnalato la possibilità di collegare il joystick in parallelo con un'altro grazie alla sua presa passante: in pratica una piccola prolunga dotata di un connettore femmina e collegata in parallelo al cavo principale del joystick. In questo modo l'utente non deve continuamente agire sulla delicata presa del computer col rischio di danneggiarla. Usando assiduamente questo futuristico joystick, abbiamo potuto scoprire che è adattissimo per applicazioni grafiche. La pratica ci ha quasi fatto entrare in simbiosi con il quadro a sensori e disegnare a video ci pare ora semplice come farlo sulla carta. Le applicazioni più soddisfacenti, comunque, rimangono nell'ambito dei videogame. In particolare abbiamo rilevato una grande predisposizione per quei giochi in cui il personaggio deve muoversi in tutte le direzioni su un'area vista dall'alto (come molti videogame sportivi di squadra, per esempio calcio, rugby, football americano, basket oppure giochi di guerra, come Rambo, Who Dares Wins e tantissimi altri). Per i videogame spaziali gioca nettamente a favore la velocità di risposta del sistema a sensori.

Il joystick Flash Fire a sensori, dunque, è sicuramente una novità che contribuisce a migliorare le prestazioni del vostro computer Commodore o non Commodore essendo, infatti, molto affidabile e compatibile con i maggiori sistemi di home computer. La stessa funzione è svolta efficacemente anche da tutti gli altri accessori prodotti dalla Gp Elettronica come il kit per la manutenzione del registratore a cassette. Chi possiede un C64 dotato del solo registratore sa quanto possa diventare fastidioso caricare un programma se questo non è in perfette condizioni. È facile per esempio iniziare a caricare un programma e vedere interrotta la procedura di caricamento dopo lunghi e noiosi minuti, con il risultato che si deve ricominciare da capo.

I motivi per cui un registratore non funziona correttamente sono essenzialmente tre: può avere la testina sporca, magnetizzata oppure disallineata. La testina è quasi sempre la responsabile del cattivo funzionamento di un registratore. Se è sporca è molto facile pulirla servendosi di uno straccetto morbido e pulito imbevuto di alcool. Se è magnetizzata, il che accade dopo molte ore di uso del registratore, specialmente se usate nastri di cattiva qualità, è necessario servirsi di un demagnetizzatore. Il difetto più grave comunque resta il disallineamento, che è anche il difetto più subdolo, poiché capita di poter leggere facilmente i file scritti con lo stesso registratore, mentre è impossibile leggere i file scritti con un altro registratore, o i nastri in commercio. Se si procede troppo tardi al riallineamento c'è il rischio di non essere più in grado di leggere i propri programmi scritti con la testina disallineata. È dunque importante avere sempre il registratore a posto, e a questo scopo ci si può servire del Super Allineatore Elettronico della Gp Elettronica. La confezione comprende un piccolo accessorio da inserire tra il registratore e il computer, una cassetta e un cacciavite antistatico da utilizzare per ripristinare il corretto azimuth della testina, oltre naturalmente alle istruzioni per l'uso. Allineare le testine con il dispositivo in questione è un'operazione molto semplice: si inserisce l'accessorio di controllo tra il registratore e il computer, si mette la cassetta in dotazione nel registratore e si preme il tasto Play. Se la testina del registratore è ben allineata il led presente sull'allineatore si accende. Se non si accende oppure lampeggia, si deve agire con il cacciavite sull'apposita vite del registratore finché

non si ottiene una perfetta accensione. Un'operazione davvero semplice e molto utile. Un altro accessorio curioso è il prova joystick che serve per controllare l'efficenza di un joystick: funziona con una pila da 9 volt e dispone di cinque led tramite i quali è possibile sincerarsi del funzionamento dei cinque contatti del joystick. È molto bello invece il joystick trasparente: si tratta di un joystick normalissimo a contatti, tuttavia è realizzato in plastica trasparente che gli fornisce un aspetto futuristico. Per chi ama il classico è comunque disponibile lo stesso modello di joystick realizzato in plastica nera. Ci congediamo da voi ricordandovi le paddle, anch'esse di buona fattura, dotate di un simpatico astuccio, utili in giochi come il mitico Arkanoid e cloni vari e tutti i piccoli accessori della vasta gamma della Gp Elettronica come cavi di prolunga, sdoppiamento e connessione varia per il vostro affezionato home.

Tutti gli oggetti sopracitati sono prodotti da Gp Elettronica, via 4 Novembre 32/34, 20092 Cinisello Balsamo (MI), tel. 02-6189551, Fax 66012023 e sono disponibili presso Electronic Performance, via S. Fruttuoso 16/A, Monza (S. Fruttuoso), tel 039/744164

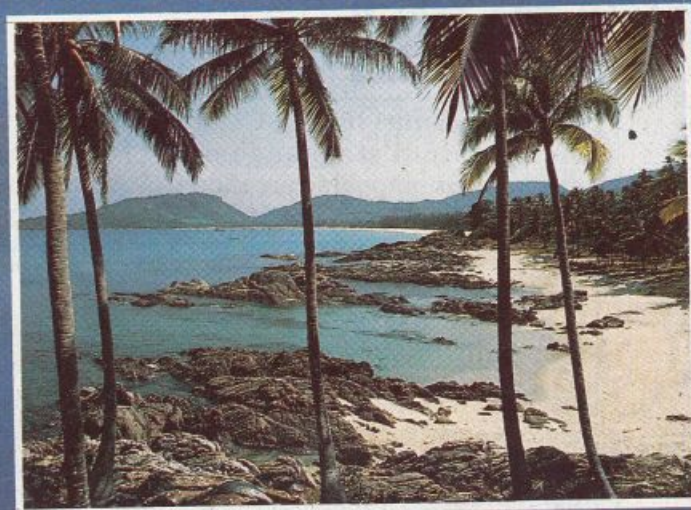
Con gli strumenti di cui siamo diventati padroni in queste ultime puntate, siamo ormai maestri di Logo. Ora possiamo cominciare a divertirci. Con questo numero impariamo, passo per passo, a creare un'avventura gestita dal computer e a lasciarci trasportare dalla fantasia nel fantastico mondo dei giochi di ruolo

Logo adventure

Sesta puntata

Capitolo quindicesimo

La massiccia diffusione dei personal computer è stata accompagnata da un'intensa produzione di giochi utilizzabili su di essi. Oltre alle trasposizioni dei vari giochi da bar, dalle sparatorie contro i mostri spaziali fino alle simulazioni dei giochi olimpici, sono divenuti popolari i cosiddetti adventure game, o giochi di avventura.

Si tratta di giochi per così dire più intelligenti degli altri, nel senso che puntano sul ragionamento e sulla fantasia del giocatore, anziché, semplicemente, sui suoi riflessi o sulla sua mira. Il giocatore di un'avventura si trova solitamente immedesimato nell'eroe di una storia, e può prendere con la massima autonomia le decisioni riguardo alle successive azioni da compiere, comunicando al computer le opportune istruzioni. Il miglior risultato si ottiene quando il computer può ricevere queste istruzioni nella lingua del giocatore; a tutt'oggi però la maggior parte dei giochi di avventura in commercio

usa come lingua l'inglese. In questo capitolo e nei due successivi, vedremo come costruire in Logo un piccolo gioco di avventura con cui si possa comunicare in italiano. La storia scelta (un naufrago che cerca un tesoro su un'isola sconosciuta) non avrà particolare importanza nel programma: cercheremo di creare uno scheletro di procedure utilizzabile in generale, che potremo poi riempire con qualunque trama ci venga in mente.

La mappa

Nel nostro gioco avremo una gamma piuttosto ristretta di azioni possibili: potremo andarcene allegramente in giro per l'isola secondo i punti cardinali, raccogliendo gli oggetti che tro-

veremo nei vari luoghi. Cominciamo quindi stendendo una schematica mappa e cercando il modo di rappresentarla in Logo. La mappa che useremo sarà quella mostrata in **figura 1**.

Ci sono in tutto sei luoghi, ognuno dei quali è contraddistinto da un numero. La particolare disposizione dei numeri non ha alcuna importanza (nel senso che potremmo usarne una qualunque), ma ci sarà molto utile averne stabilita una: potremo chiamare i luoghi con il loro numero anziché per nome, e questo, come vedremo, semplificherà parecchio le cose. Ma vediamo come si svolgerà il gioco. Dopo una breve introduzione per illustrare lo scenario, il nostro giocatore si troverà in mezzo a una misteriosa jungla (locazione 1).

Se tenta di dirigersi verso nord (locazione 2) sarà immediatamente fulminato dal dio che domina il tempio, mentre se va verso sud (locazione 4) sarà assalito dai cannibali. Egli naturalmente non avrà la mappa sott'occhio, ed è destinato ad apprendere tutto questo a proprie spese (quando si muore, il gioco riparte dall'inizio).

Il dio potrà essere placato soltanto portandogli la statuetta che si trova nel villaggio dei cannibali, e i cannibali a loro volta saranno innocui se verrà portata loro la conchiglia che si trova sulla spiaggia (locazione 5). Sarà quindi possibile per il nostro eroe andarsene per prima cosa sulla spiaggia, prendere la conchiglia, portarla ai cannibali, e lì prendere la statuetta da offrire al dio.

Quando raggiungerà la caverna del tesoro (locazione 3) sarà immediatamente assalito da un ferocissimo orso, e non riuscirà a spuntarla a meno che non abbia precedentemente raccolto la spada che si trova nell'arsenale abbandonato (locazione 6). A questo punto, ormai allo stremo delle forze, egli potrà finalmente impossessarsi dell'agognato tesoro.

Come possiamo dunque rappresentare questa mappa in Logo? Beh, ci serviremo di tabelle scritte in forma di liste.

Il modo più funzionale è probabilmente quello di associare a ogni locazione una tabella di quattro elementi, uno per ciascuno dei punti cardinali. Per esempio, la tabella della locazione di partenza, cioè la jungla (numero 1) sarà fatta come in **tavola 1**. Che vuol dire: verso nord c'è la locazione 2 (che è il tempio), verso est non c'è niente (la locazione 0 non esiste), verso sud c'è la locazione 4 (villaggio dei cannibali) e verso ovest c'è la spiaggia (numero 5). Facendo lo stesso per tutti e sei i luoghi della nostra mappa, otteniamo lo schema riportato in **tavola 2**.

Possiamo tradurre direttamente questa trasposizione della mappa in una lista di liste, che chiameremo appunto Mappa.

```
AS "MAPPA
[[2 0 4 5]
[0 3 1 0]
[0 0 0 2]
[1 0 0 6]
[0 1 0 0]
[0 4 0 0]]
```

La lista è stata incolonnata per chiarezza, ma digitandola sul computer non sarà possibile andare a capo con <Return> prima della fine dell'istruzione; al massimo potrete lasciare degli spazi.

Supponiamo ora di essere nella locazione 1 e di tenere questo valore in una variabile globale:

```
AS "DOVESONO 1
```

Ora per avere la tabella relativa a quella locazione basta considerare:

```
ELEMENTO :DOVESONO :MAPPA
```

Questo ci riporterà la lista [2 0 4 5]. Se vogliamo spostarci, per esempio, a sud ci basterà prendere il terzo elemento ed Assegnarlo a Dovesono.

Non ci darebbe molta soddisfazione a sentirci rispondere dal computer qualcosa come "adesso siamo nella locazione uno". Prepariamo quindi un'altra lista che contenga, in ordine, le descrizioni dei vari luoghi.

Tavola 1.

Schema numerico della locazione 1 (Jungla)

N	*E*	*S*	*O*
2	*0*	*4*	*5*

Tavola 2.

Schema numerico di tutte le locazioni della mappa

Luogo	*N*	*E*	*S*	*O*
1 (jungla)	*2*	*0*	*4*	*5*
2 (tempio)	*0*	*3*	*1*	*0*
3 (caverna)	*0*	*0*	*0*	*2*
4 (villaggio)	*1*	*0*	*0*	*6*
5 (spiaggia)	*0*	*1*	*0*	*0*
6 (arsenale)	*0*	*4*	*0*	*0*

Dovrete usare l'editore per inserirla: è troppo lunga per un comando immediato.

```
AS "DESCRIZIONI [[SEI IN UNA INTRICATISSIMA JUNGLA.]
[SEI DI FRONTE ALL'IMPONENTE TEMPIO
DEL DIO DELL'ISOLA.]
[SEI IN UNA OSCURA CAVERNA SULLE
PENDICI DEL VULCANO.]
[SEI NEL BEL MEZZO DEL VILLAGGIO
DEI CANNIBALI.]
[SEI SU UNA SPIAGGIA SOLEGGIATA.]
[SEI NEI PRESSI DI UN ARSENALE
ABBANDONATO DAI PIRATI.]]
```

Ora, per ricevere una esauriente descrizione del luogo in cui ci troviamo, ci basterà digitare:

```
ST ELEMENTO :DOVESONO :DESCRIZIONI
```

Scriviamo subito una procedura che faccia questo:

```
PER GUARDA
ST ELEMENTO :DOVESONO :DESCRIZIONI
FINE
```

Provate a cambiare il valore di Dovesono e a digitare ogni volta Guarda: comincerete piano piano ad avere la sensazione di aggirarvi veramente sull'isola.

La procedura principale

La procedura che guiderà il nostro programma sarà piuttosto semplice: essa dovrà ricevere un comando (letto con Lr), verificare se è scritto correttamente (Interpretarlo) e poi Eseguirlo. Infine controllerà se il gioco è finito.

Se lo è chiamerà una apposita procedura Haivinto; altrimenti richiamerà se stessa per ricominciare il ciclo.

Tempio 2 | Caverna 3
Spiaggia 5 | Jungla 1
Arsenale 6 | Villaggio 4

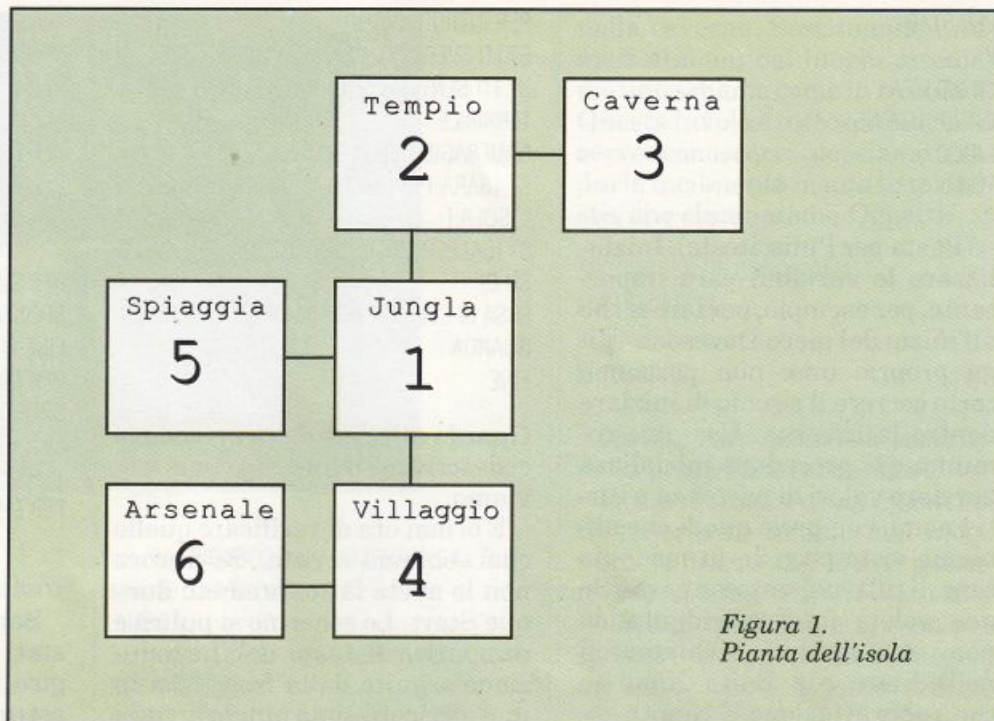



Figura 1. Pianta dell'isola

```
PER GIOCO
ST [E ADESSO?]
ESEGUI INTERPRETA LR
SE FINITO? ALLORA HAIVINTO
GIOCO
FINE
```

Penseremo più avanti al compito specifico di Interpreta; dovrà comunque essenzialmente riportare la stessa lista letta da Lr, dopo avere controllato che non contenga errori.

Per ora, utilizzando un vecchio trucco della programmazione, facciamo finta di averla già definita, così la procedura Gioco funzionerà lo stesso e potremo provarla mentre proseguiamo.

Che cosa significa "facciamo finta"?

Significa che possiamo definire una procedura Interpreta che non fa proprio niente; quando ne avremo il tempo provvederemo a sostituirla con una procedura seria.

```
PER INTERPRETA :COMANDO
RI :COMANDO
FINE
```

Facciamo ora la stessa cosa con Finito?, facendogli riportare sempre Falso.

```
PER FINITO?
RI "FALSO
FINE
```

Con questo, durante l'esecuzione di Gioco non si dovrebbero incontrare procedure non definite.

Esegui è una primitiva: vuole una lista di comandi Logo come argomento e non fa altro che eseguirla.

Per poterci capire, Esegui [Guarda] funziona esattamente come Ripeti 1 [Guarda].

Ovviamente se la procedura Guarda non esiste così facendo si otterrà sicuramente un messaggio di errore.

L'inizio del gioco

Per poter dare inizio a una partita, avremo bisogno di una procedura che stampi sullo schermo una brevissima presentazione del nostro gioco e che, soprattutto, inizializzi le variabili:

```
PER START
PT
INIZIALIZZA
INTRODUZIONE
GIOCO
FINE
```

(Pt sta per Puliscitesto). Inizializzare le variabili sarà importante, per esempio, per fare sì che all'inizio del gioco Dovesono valga proprio uno: non possiamo certo correre il rischio di iniziare dentro la caverna. Useremo comunque la procedura Inizializza per dare valori di partenza a tutti i nomi, compresi quelli che abbiamo visto poco fa, in modo da tenerli tutti nello stesso posto. Se non volete faticare ridigitando nomi già definiti, richiamateli nell'editore con Edita Nomi (o con l'abbreviazione E Nomi):

```
PER INIZIALIZZA
AS "DOVESONO 1AS "MAPPA
[[2 0 4 5]
[0 3 1 0]
[0 0 0 2]
[1 0 0 6]
[0 1 0 0]
[0 4 0 0]]
AS "DESCRIZIONI [[SEI IN UNA INTRICATISSIMA JUNGLA.]
[SEI DI FRONTE ALL'IMPONENTE TEMPIO DEL DIO DELL'ISOLA.]
[SEI IN UNA OSCURA CAVERNA SULLE PENDICI DEL VULCANO.]
[SEI NEL BEL MEZZO DEL VILLAGGIO DEI CANNIBALI.]
[SEI SU UNA SPIAGGIA SOLEGGIATA.]
[SEI NEI PRESSI DI UN ARSENALE ABBANDONATO DAI PIRATI.]]
FINE
```

La procedura Introduzione sarà invece qualcosa come:

```
PER INTRODUZIONE
ST [TI SVEGLI ALL'IMPROVVISO.]
ST [TI SENTI FRASTORNATO E RAMMENTI IL TERRIBILE
NAUFRAGIO DELLA SCORSA NOTTE.]
ST [SARÀ PROPRIO IL CASO DI ESPLORARE L'ISOLA:]
ST [SAI PER CERTO CHE DA QUALCHE PARTE SU DI
ESSA SI CELA UN INESTIMABILE TESORO.]
GUARDA
FINE
```

Guarda alla fine della procedura ci descriverà il luogo in cui ci troviamo.

È ormai ora di verificare quello che abbiamo scritto. Se ancora non lo avete fatto, premete dunque Start. Lo schermo si pulirà e comparirà il testo dell'Introduzione seguito dalla frase "Sei in una intricatissima jungla", che la procedura Guarda è andata a prendersi nella lista delle descrizioni. Poi viene stampato un "E adesso?", con il quale la procedura Gioco si aspetta un comando da eseguire. Purtroppo, riguardo al gioco non abbiamo ancora molte cose da chiedere.

Come spostarsi sull'isola

Per andare da una locazione all'altra useremo i comandi Nord, Est, Sud e Ovest, che corrispondono ripettivamente al primo, al secondo, al terzo e al quarto elemento della tabella relativa a una certa locazione.

```
PER NORD
SPOSTATI 1
FINE

PER EST
SPOSTATI 2
FINE

PER SUD
SPOSTATI 3
FINE
PER OVEST
SPOSTATI 4
FINE

PER SPOSTATI :DIR
........
```

Sarà dunque la procedura Spostati a permetterci di andare in giro. Essa dovrà prima di tutto estrarre dalla mappa la tabella relativa al luogo in cui ci troviamo (cioè a Dovesono), usando:

```
ELEMENTO :DOVESONO :MAPPA
```

Questo riporterà una lista di quattro numeri, per esempio [2 0 4 5]: se stiamo andando a nord dovremo considerare l'elemento numero 1 di questa lista (e spostarci quindi, nell'esempio, nella locazione 2), per andare ad est l'elemento numero 2, e così via. Il numero sarà contenuto nel parametro Dir, passato a Spostati direttamente dalle procedure Nord, Est, eccetera. Il luogo in cui andremo a finire sarà quindi dato, nell'esempio, da:

```
ELEMENTO :DIR [2 0 4 5]
```

ovvero, in generale:

```
ELEMENTO :DIR ( ELEMENTO :DOVESONO :MAPPA )
```

Proseguendo nella scrittura di Spostati, mettiamo

Tavola 3.

Rappresentazione numerica degli oggetti

luogo	*oggetto*
5	*CONCHIGLIA*
4	*STATUETTA*
6	*SPADA*
3	*TESORO*

provvisoriamente questo valore in una variabile locale:

```
PER SPOSTATI :DIR
LOCALE "NUOVOLUOGO
AS "NUOVOLUOGO ELEMENTO :DIR ( ELE-
MENTO :DOVESONO
:MAPPA )
```

Ora Nuovoluogo contiene il numero del luogo in cui ci trasferiremo camminando nella direzione scelta. Potrebbe, però, anche contenere zero: con zero avevamo voluto indicare che una certa direzione è preclusa. In questo caso sarà opportuno stampare un messaggio e, ovviamente, non eseguire alcuno spostamento.

```
PER SPOSTATI :DIR
LOCALE "NUOVOLUOGO
AS "NUOVOLUOGO ELEMENTO :DIR ( ELE-
MENTO :DOVESONO
:MAPPA )
SE :NUOVOLUOGO = 0 ST [NON SI PUO ANDA-
RE DA QUELLA
PARTE!] STOP
```

Altrimenti, per effettuare lo spostamento ci basterà cambiare il valore di Dovesono e informare il giocatore della sua nuova situazione chiamando Guarda:

```
PER SPOSTATI :DIR
LOCALE "NUOVOLUOGO
AS "NUOVOLUOGO ( ELEMENTO :DIR ELE-
MENTO :DOVESONO
:MAPPA )
SE :NUOVOLUOGO = 0 ST [NON SI PUO ANDA-
RE DA QUELLA
PARTE] STOP
ST "OK.
AS "DOVESONO :NUOVOLUOGO
GUARDA
FINE
```

Con questo, le procedure Nord, Sud, Est e Ovest dovrebbero fare il loro dovere. Lanciate Start e divertitevi a esplorare l'isola.

Non sarebbe male associare a ogni luogo il relativo disegno eseguito con la tartaruga. Se volete specializzarvi in paesaggi, preparate quelli necessari e conservateli con Conserdis. Potrete poi inserire il relativo Recupdis nella procedura Guarda.

Capitolo sedicesimo

Come rappresentare gli oggetti

Ora che abbiamo uno spazio in cui è possibile muoversi, cominciamo a studiare il modo di lasciare degli oggetti nei vari luoghi. In particolare, nel nostro gioco gli oggetti saranno quattro: una conchiglia sulla spiaggia, una statuetta nel villaggio, una spada nell'arsenale e il tesoro nella caverna. Sostituendo i numeri ai nomi dei luoghi otteniamo uno schema come in **tavola 3**. Questa tavola è tutto quello che ci serve conoscere; possiamo tradurla facilmente in una lista di liste, che chiameremo Oggetti:

```
AS "OGGETTI [[5 CONCHIGLIA]
[4 STATUETTA]
[6 SPADA]
[3 TESORO]]
```

Questa istruzione va inserita nella procedura Inizializza.

Ora per sapere quali oggetti si trovano, per esempio, nel luogo numero 6, sarà sufficiente passare in rassegna tutta la lista alla ricerca di quel numero; così, per spostare un oggetto da un luogo a un altro, basterà sostituirne il vecchio numero con il nuovo.

Lo stesso sistema ci permetterà di raccogliere oggetti e di portarceli dietro: basterà stabilire che quando a un oggetto è associato un certo numero (noi useremo -1) esso è in mano al giocatore.

Poiché appena giunti in un luogo sarà importante sapere quali oggetti vi si trovano, facciamo in modo che la procedura Guarda scritta poco fa provveda anche a informarci di questo:

```
PER GUARDA
ST ELEMENTO :DOVESONO :DESCRIZIONI
ST "VEDI:
STAMPAOGGETTI :DOVESONO
FINE
```

Stampaoggetti dovrà quindi scrivere sullo schermo i nomi degli oggetti che si trovano nel luogo indicato dal suo parametro; faremo in modo che quando non trova nessun oggetto scriva un messaggio particolare:

```
PER STAMPAOGGETTI :LUOGO
LOCALE "TROVATI
AS "TROVATI COSAC'È :LUOGO :OGGETTI []
SE VUOTO? :TROVATI ST [UN BEL NIENTE!]
STOP
```

```
ST :TROVATI
FINE
```

La procedura Cosac'è, che vedremo subito, riporterà la lista degli oggetti che si trovano nel Luogo; Stampaoggetti mette questa lista nella variabile locale Trovati e poi, se non è vuota, la stampa. La procedura più spinosa sarà proprio questa Cosac'è:

```
PER COSAC'È :LUOGO :TABELLA :LISTA
```

I tre parametri rappresentano rispettivamente: il Luogo (sotto forma di numero) su cui vogliamo informazioni; la Tabella degli oggetti (che passiamo come parametro per poterla esaminare in modo ricorsivo); e infine la Lista degli oggetti trovati.

Quest'ultima lista verrà riempita passo per passo: quando chiamiamo la procedura essa deve essere vuota.

Abbiamo detto che agiremo ricorsivamente sulla tabella.

Come di consueto, occupiamoci prima di tutto della condizione di base riguardante la lista vuota: quando la Tabella è vuota gli oggetti che ci interessano dovranno già essere stati messi nella Lista, che possiamo quindi senz'altro riportare:

```
PER COSAC'È :LUOGO :TABELLA :LISTA
SE VUOTO? :TABELLA RI :LISTA
```

Ora pensiamo all'altro caso, quello cioè in cui non siamo ancora arrivati in fondo.

Consideriamo il primo elemento della tabella: esso sarà una lista di due elementi, un numero e una parola (come [4 Statuetta]), che chiameremo Coppia.

```
PER COSAC'È :LUOGO :TABELLA :LISTA
SE VUOTO? :TABELLA RI :LISTA
LOCALE "COPPIA
AS "COPPIA PRI :TABELLA
```

Prendiamo il primo elemento della coppia: se esso è uguale al Luogo dovremo aggiungere alla Lista l'oggetto relativo (che è l'ultimo elemento della coppia).

```
PER COSAC'È :LUOGO :TABELLA :LISTA
SE VUOTO? :TABELLA RI :LISTA
LOCALE "COPPIA
AS "COPPIA PRI :TABELLA
SE PRI :COPPIA = :LUOGO AS "LISTA ( FR
:LISTA
 ULT :COPPIA )
```

Ed ora, la chiamata ricorsiva: fai la stessa cosa sul resto della tabella:

```
PER COSAC'È :LUOGO :TABELLA :LISTA
SE VUOTO? :TABELLA RI :LISTA
LOCALE "COPPIA
AS "COPPIA PRI :TABELLA
SE PRI :COPPIA = :LUOGO AS "LISTA ( FR :LISTA
ULT :COPPIA )
RI COSAC'È :LUOGO MP :TABELLA :LISTA
FINE
```

Come raccogliere gli oggetti

La procedura Stampaoggetti è stata scritta in modo abbastanza generale da poterci dare informazioni su un qualsiasi luogo, indipendentemente da dove ci troviamo: tra poco potremo utilizzarla per fare l'inventario delle cose che abbiamo con noi.

```
PER PRENDI :OGG
SE NON C'È? :OGG :DOVESONO ST [NON C'È!]
STOP
ST "OK.
SPOSTA :OGG :DOVESONO (-1)
FINE
```

Questa è la procedura che ci permetterà di raccogliere spade e tesori. Esaminiamola riga per riga. Prima di tutto dobbiamo verificare che l'oggetto che vogliamo prendere sia effettivamente nel luogo in cui ci troviamo, indicato in Dovesono.

Per la verifica utilizziamo il predicato C'è? che scriveremo subito: se per caso l'oggetto Non c'è non facciamo proprio niente. In caso contrario scarichiamo il lavoro alla procedura Sposta, che avrà tre argomenti: l'oggetto, la locazione di partenza e quella di destinazione. Per esempio, l'ultima riga di Prendi si potrà leggere così: Sposta l'Oggetto dalla locazione Dovesono alla locazione -1 (cioè a me). Ma pensiamo a definire C'è?. Sicuramente, un certo oggetto sarà in un luogo solo se nella tabella chiamata Oggetti c'è una lista che contiene il luogo e l'oggetto stessi. In altre parole, per sapere se la Spada è nella locazione 5 basterà guardare se la lista [5 Spada] appartiene a Oggetti.

```
PER C'È? :OGG :LUOGO
RI AP? ( LISTA :LUOGO :OGG ) :OGGETTI
FINE
```

Pensiamo adesso a come definire Sposta. Qualcuno ricorderà la procedura Sosti che era stata proposta alcuni capitoli fa. Essa avrebbe dovuto essere scritta più o meno così:

```
PER SOSTI :NUOVO :VECCHIO :LISTA
SE VUOTO? :LISTA RI :LISTA
SE PRI :LISTA = :VECCHIO RI INPRI :NUOVO MP
:LISTA
RI INPRI ( PRI :LISTA ) SOSTI :NUOVO :VEC-
CHIO MP :LISTA
FINE
```

Questa procedura prende una Lista, vi sostituisce l'oggetto Vecchio (se lo trova) con il Nuovo, e riporta il risultato della sostituzione. Essa è tutto quello che ci serve per definire Sposta: dovremo soltanto cambiare il valore di Oggetti, Sostituendo la coppia che riguarda l'oggetto che dobbiamo spostare:

```
PER SPOSTA :OGG :DA :A
AS "OGGETTI SOSTI ( LISTA :A :OGG ) ( LISTA
:DA :OGG )
:OGGETTI
FINE
```

Per esempio, se vogliamo prendere la spada in locazione 6, Prendi chiamerà Sposta "Spada 6 (-1), e Sosti sostituirà [-1 Spada] a [6 Spada]. Scrivere una procedura per lasciare un oggetto sarà ora piuttosto facile: basterà spostare quell'oggetto dal luogo -1 al luogo Dovesono.

```
PER LASCIA :OGG
SE NON C'È? :OGG (-1) ST [NON CE L'HAI!]
STOP
ST "OK.
SPOSTA :OGG (-1) :DOVESONO
FINE
```

Ci servirà anche una procedura che faccia l'inventario di quello che stiamo portando. Si tratta di guardare quali oggetti si trovano in un certo luogo (-1), e possiamo quindi semplicemente utilizzare Stampaoggetti:

```
PER INVENTARIO
ST [HAI CON TE:]
STAMPAOGGETTI (-1)
FINE
```

Un piccolo trucco

Lanciate Start: ora oltre ad andarvene a zonzo per tutta l'isola potrete anche far eseguire, per esempio, Prendi "Conchiglia o Lascia "Statuetta.

Le virgolette, però, sono scomode, ed escogitiamo un brillante stratagemma per poterne fare a meno.

Visto che il Logo tratta come procedure tutte le parole senza virgolette, definiremo per ogni oggetto una procedura che riporta il suo stesso nome.ln1:

```
PER CONCHIGLIA
RI "CONCHIGLIA
FINE

PER STATUETTA
RI "STATUETTA
FINE

PER SPADA
RI "SPADA
FINE

PER TESORO
RI "TESORO
FINE
```

Ora possiamo digitare tranquillamente frasi come Lascia Tesoro, senza virgolette di sorta. Nel prossimo capitolo faremo un altro passo avanti, rendendo comprensibili frasi come Prendi Quella orribile statuetta, Guarda meglio stupido, e altre simili.

Lanciatevi pure in esplorazione. Se avete raccolto un oggetto, potrete controllare l'Inventario per verificare che lo state proprio portando con voi.

Dopo un po' di questo vagabondare deciderete probabilmente che in fondo, tutto questo, non è molto emozionante.

Nel prossimo paragrafo lavoreremo moltissimo per poter inserire all'interno del gioco i pericoli di cui abbiamo parlato sin dall'inizio.

Tavola 4.

Rappresentazione schematica dei pericoli dell'avventura

luogo	*pericolo*	*condizione di vita*
locazione	*1*	*[ ]*
locazione	*2*	*dio avere statuetta*
locazione	*3*	*orso avere spada*
locazione	*4*	*cannibali avere conchiglia*
locazione	*5*	*[ ]*
locazione	*6*	*[ ]*

E adesso viene il bello

Riassumiamo il funzionamento del nostro programma: dopo la procedura di lancio, Start, tutto avviene nell'ambito della procedura Gioco: essa riceve un comando, lo esegue e controlla (o meglio controllerà) se la partita è finita; se non lo è, ricomincia.

I comandi validi ai fini del gioco sono i seguenti:

- Guarda;
- Inventario;
- Prendi un oggetto;
- Lascia un oggetto;
- Nord, Sud , Est, Ovest.

Ed ecco i pericoli che inseriremo:

- Se entriamo nel villaggio dei cannibali senza la conchiglia, moriremo mangiati. Altrimenti, se abbiamo la conchiglia con noi, sopravviveremo e diventeremo re del villaggio.

- Se ci rechiamo al tempio senza la statuetta, moriremo fulminati. Se invece l'abbiamo saremo risparmiati.

- Se entriamo nella caverna senza la spada, moriremo sbranati. Altrimenti la scampiamo.

Lo schema è sempre lo stesso, e il rischio si corre solo durante uno spostamento. Deputiamo quindi la procedura Spostati ad accertarsi dell'eventuale pericolo e, eventualmente, a eseguirlo. Basterà aggiungere una riga in fondo:

```
PER SPOSTATI :DIR
LOCALE "NUOVOLUOGO
AS "NUOVOLUOGO ELEMENTO :DIR ELEMEN-
TO :DOVESONO :MAPPA
SE :NUOVOLUOGO = 0 ST [NON SI PUO ANDA-
RE DA
QUELLA PARTE!] STOP
AS "DOVESONO :NUOVOLUOGO
GUARDA
SE C'ÈPERICOLO? ALLORA ESEGUIPERICOLO
FINE
```

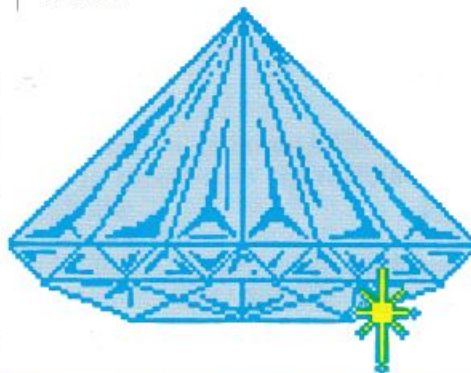
Ma come rappresentiamo questi pericoli? Usiamo una tabella (**tavola 4**). Notiamo subito che la seconda colonna, scritta così, non è rilevante per il programma. Qualunque sia il pericolo, l'esecuzione non cambia: Se si verifica la condizione di vita, Allora si sopravvive, Altrimenti si muore. Naturalmente cambieranno i messaggi da stampare sullo schermo nei vari casi.

Traduciamo per ora soltanto la terza colonna in una lista, utilizzando il predicato C'è? definito poco fa (***n.b***: questa linea va inserita in Inizializza).

```
AS "CONDVITA [[]
[C'È? STATUETTA (-1)]
[C'È? SPADA (-1)]
[C'È? CONCHIGLIA (-1)]
[]
[]]
```

Inserire le liste vuote è necessario: se il primo elemento della lista fosse [C'è? "Statuetta (-1)] dovremmo credere che quel pericolo riguarda la locazione 1, cioè la jungla. Guardando questa lista salta agli occhi un modo in cui potremmo scrivere il predicato C'èpericolo? che abbiamo inserito in Spostati: basterà accertarsi che al posto della Condizione di Vita relativa al luogo in cui ci troviamo non ci sia la lista vuota:

```
PER C'ÈPERICOLO?
RI NON VUOTO? ELEMENTO :DOVESONO
:CONDVITA
FINE
```

A questo punto Spostati è in grado, quando è il caso, di chiamare la procedura Eseguipericolo. Essa dovrà per prima cosa sapere se il nostro destino è la vita o la morte. A questo scopo definiamo subito un apposito predicato:

```
PER SOPRAVVIVI?
RI ESEGUI ELEMENTO :DOVESONO :CONDVITA
FINE
```

Ovvero, riporta il risultato della Condizione di Vita relativa a questo luogo. Quest'ultima sarà qualcosa come [C'è? Statuetta (-1)] e quando viene Eseguita riporterà Vero (nel caso si sopravviva) o Falso (nel caso si muoia).

In base alla risposta di Sopravvivi? la procedura Eseguipericolo saprà sicuramente cosa scegliere:

```
PER ESEGUIPERICOLO
SE SOPRAVVIVI? VIVI ALTRIMENTI MUORI
FINE
```

Tutto quello che ci resta da definire sono dunque le procedure Vivi e Muori. Entrambe dovranno stampare dei messaggi particolari per informare il giocatore di quello che sta succedendo, e ogni locazione dovrà avere il suo proprio messaggio. Procediamo quindi nel solito modo:

```
AS "MESSAGGIVITA [[]
[HAI RIPORTATO AL TEMPIO LA SACRA
STATUETTA RUBATA SECOLI FA DAI
CANNIBALI. IL DIO È RICONOSCENTE.]
[L'ORSO CHE ABITA LA CAVERNA TI
ATTACCA, MA TU LO TRAFIGGI CON
UN PRECISO COLPO DI SPADA.]
[LA CONCHIGLIA CHE HAI CON TE È
IL MAGICO SIMBOLO DELLA TRIBU,
PERDUTO DA GENERAZIONI.
I CANNIBALI TI PROCLAMANO LORO RE.]
[]
[]]
```

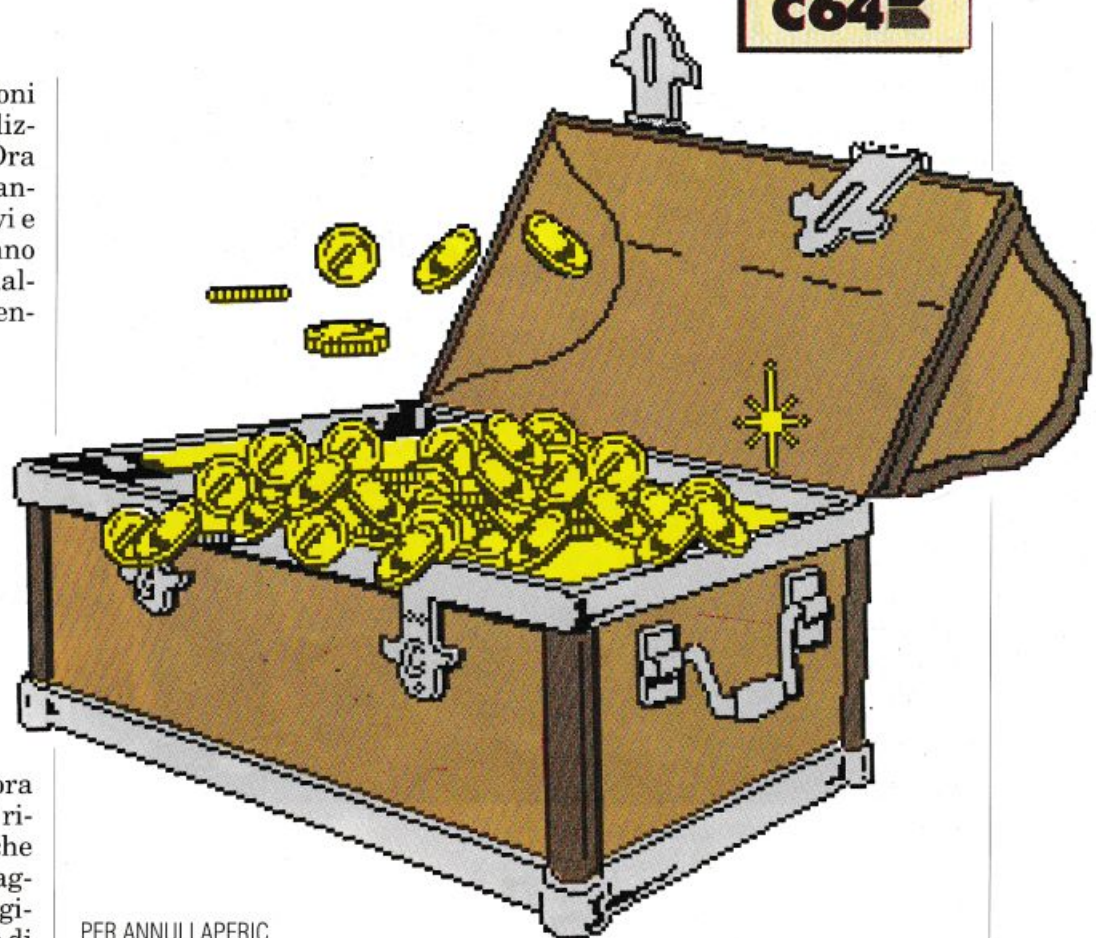
```
AS "MESSAGGIMORTE [[]
[HAI PROFANATO IL TEMPIO DEL
POTENTISSIMO DIO, CHE TI FULMINA
CON LE SUE SAETTE.]
[L'ORSO CHE ABITA LA CAVERNA TI
ATTACCA FEROCEMENTE. NON HAI ARMI
PER DIFENDERTI.]
  ALCUNE DECINE DI CANNIBALI TI SALTANO
  ADDOSSO E TI METTONO IN PADELLA.]
  []
  []]
```

Anche queste due istruzioni andrebbero inserite in Inizializza (ma non è indispensabile). Ora abbiamo un'idea di come dovranno funzionare le procedure Vivi e Muori: per prima cosa dovranno stampare il messaggio tratto dalla rispettiva lista e corrispondente al luogo in cui ci si trova:

```
PER VIVI
ST ELEMENTO :DOVESONO :MESSAG-
GIVITA
..........
PER MUORI
ST ELEMENTO :DOVESONO :MESSAG-
GIMORTE
..........
```

È tutto qui? Riflettiamo un attimo, e supponiamo di essere appena stati proclamati re dai cannibali: se ora usciamo dal villaggio e poi vi ritorniamo, non vorremmo che uscisse fuori di nuovo il Messaggiovita. Occorrerà quindi "registrare" in qualche modo il fatto di avere già scampato il pericolo. Chiameremo la procedura che fa questo Annullaperic e la faremo chiamare da Vivi.

```
PER VIVI
ST ELEMENTO :DOVESONO :MESSAGGIVITA
ANNULLAPERIC
FINE
```

Andiamo a definire Annullaperic. Ricordate che era la procedura Spostati a decidere se occorreva o meno eseguire un pericolo? Essa lo faceva mediante C'èpericolo?, controllando la listona delle Condizioni di vita. Se ora noi sostituiamo la lista vuota alla condizione di vita del pericolo appena scampato, il predicato C'èpericolo? penserà che tutto va bene e risponderà Falso. Dobbiamo quindi modificare la lista Convita: Sostituirvi la lista vuota alla condizione corrispondente a Dovesono (che è Elemento: N Dovesono: Condvita)

```
PER ANNULLAPERIC
AS "CONDVITA SOSTI [] ( ELEMENTO :DOVE-
SONO :CONDVITA )
:CONDVITA
FINE
```

Adesso tocca alla procedura Muori. Essa è tanto semplice che si spiega da sola:

```
PER MUORI
ST ELEMENTO :DOVESONO :MESSAGGI-
MORTE
ST [SEI MORTO!]
ST [PREMI UN TASTO PER RICOMINCIARE.]
ASPETTASTO
START
FINE

PER ASPETTASTO
RIPETI 10000 [SE TASTO? STOP]
FINE
```

Aspettasto non fa altro che aspettare un po': se viene premuto un tasto si ferma (il predicato Tasto? riporta Vero quando c'è un tasto premuto); altrimenti termina da sola quando ha finito il ciclo di Ripeti.

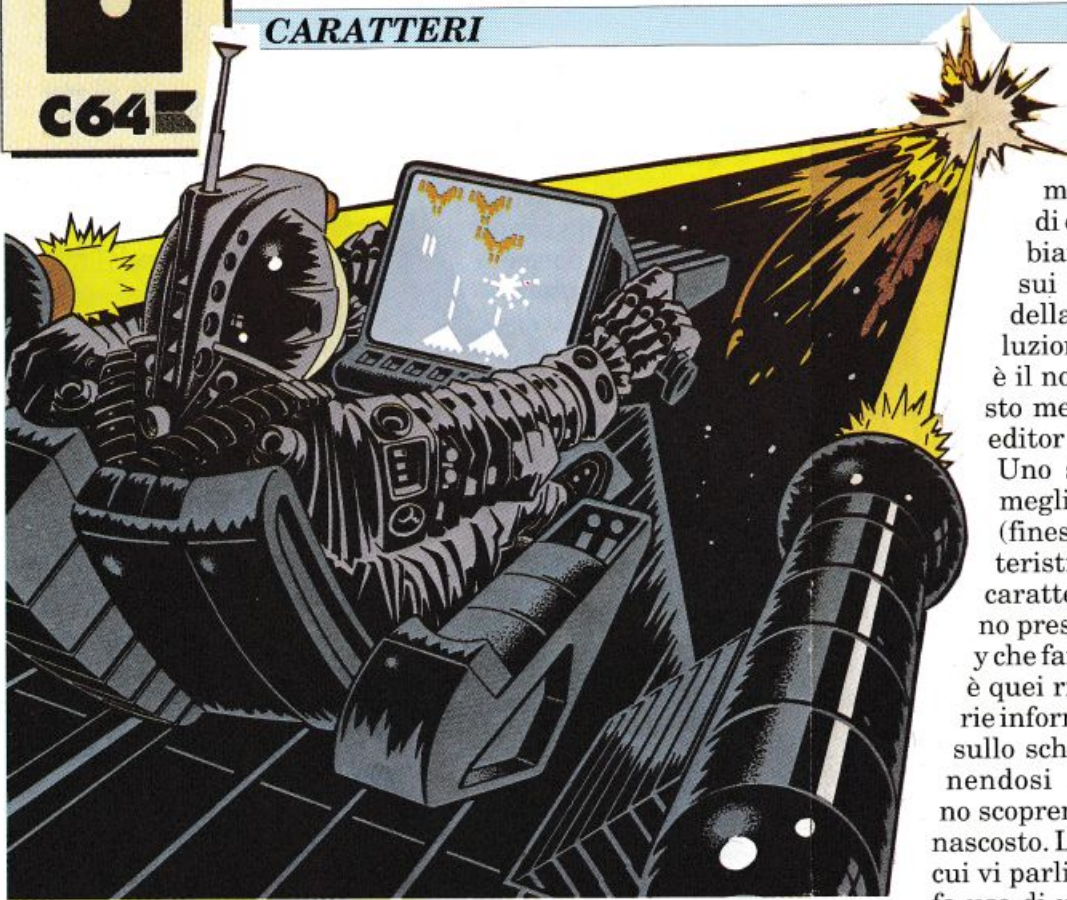
L'ultimo sforzo

Adesso funziona tutto! Manca solo la conclusione del gioco (le procedure Finito? e Haivinto che avevamo lasciato in sospeso) è presto fatto: il gioco finirà quando il tesoro è nostro cioè:

```
PER FINITO?
RI C'È? "TESORO (-1)
FINE
```

Haivinto consisterà invece di un messaggio di congratulazioni e di un arresto del programma.

```
PER HAIVINTO
ST [COMPLIMENTI, HAI VINTO!]
PUNTOACAPO
FINE
```

Luigi Ambrosi
(continua)

Sfida all'alta risoluzione

Ancora grafica. Prendendo come spunto la potentissima utility di questo mese mettiamo in luce il problema della gestione dei caratteri ridefiniti mono e multicolore e puntiamo l'attenzione sulla gestione avanzata dello scrolling caratteristico di molti videogame

Il programma di utilità che pubblichiamo questo mese è, a nostro avviso, uno strumento assolutamente necessario per tutti coloro che vogliono trarre la massima soddisfazione dall'uso amatoriale del computer. Il Commodore 64 offre prestazioni grafiche troppo allettanti e ogni vero appassionato non può essere soddisfatto se non ne conosce tutti i segreti. Nello spiegare minuziosamente tutte le funzioni di questo programma abbiamo cercato di far luce sui problemi di gestione della grafica in bassa risoluzione. Char Work, questo è il nome dell'utility di questo mese, è un potentissimo editor di caratteri e schermi. Uno schermo, che sarebbe meglio chiamare window (finestra) date le sue caratteristiche, è una griglia di caratteri. Molti di voi avranno presente le svariate utility che fanno uso di window, cioè quei riquadri contenenti varie informazioni che compaiono sullo schermo, come sovrapponendosi a esso, e scompaiono scoprendo quello che avevano nascosto. Lo stesso programma di cui vi parliamo in questa rubrica fa uso di una window quando si preme il tasto F7.

Una window, però, non è necessariamente più piccola dello schermo del nostro piccolo computer. Tutti avrete presente quei videogame caratterizzati da uno sfondo in scrolling (scorrimento). Moltissimi giochi spaziali fanno uso di questa tecnica per dare l'impressione del movimento dell'astronave e del paesaggio di grandi dimensioni. Molti giochi, addirittura, essendo in grado di gestire uno scrolling in tutte le direzioni, danno l'impressione di territori varie volte più grandi dello schermo (che è una griglia di 40 X 25 caratteri). Teoricamente, con l'aiuto di Char Works e di un po' di esperienza di programmazione in linguaggio macchina, è possibile creare gli stessi effetti dei videogame sopracitati. Noi ci limiteremo a spiegare come gestire dal Basic una window grande come lo schermo, sia in modo standard, sia in modo multicolore, nella speranza che questo invogli il lettore a sfruttare più a fondo le potenzialità del fantastico C64.

Il programma

Il programma si trova sulla cassetta allegata alla rivista e si chiama Char Works. Caricatelo normalmente con Load e lanciatelo con il comando Run. La schermata principale del programma (**figura 1**) è divisa in due parti. Nella parte superiore si vede la sagoma ingrandita del carattere "a" all'interno di un riquadro (riquadro carattere). Sulla destra ci sono alcune informazioni relative al set di caratteri e alla window. Nella parte bassa dello schermo si vede una porzione della window vuota e individuata da un bordo di asterischi.

Per scrivere qualcosa sulla window, salvarla su disco e visualizzarla da un programma Basic, seguite passo per passo le istruzioni del prossimo paragrafo. Se invece vi sentite già in grado di destreggiarvi passate direttamente al paragrafo successivo. Costruiamo una window raffigurante la scritta Char Work in una piccola cornice (**figura 2**). Premete il tasto J. In questo modo il cursore lampeggiante che prima si trovava nel riquadro carattere in alto a sinistra si sposta nella parte bassa dello schermo all'interno della window vuota. Agendo sui tasti cursore muovete la window fino a quando non vi trovate ad avere solo una riga orizzontale di asterischi appena sotto il riquadro carattere. In questo momento il cursore si trova alla riga 9 e alla colonna 21 della window. Con l'aiuto dei tasti di addizione (+) e sottrazione (-) portate l'indicatore carattere (si trova in alto e in centro sullo schermo di lavoro) fino al valore $40. Ricordate che tutti gli indicatori di questo programma mostrano numeri in formato esadecimale (ma questo non dovrebbe creare problemi per il momento; date un'occhiata, comunque, alla **tavola 1**). Ora premete il tasto asterisco (*) per fissare sulla window il carattere selezionato. Basandovi sul punto di riferimento così fissato e muovendo la window con il cursore, piazzate lo stesso carattere sulla stessa linea per sette spazi a destra e otto a sinistra, rispetto al punto di riferimento. Ora sarete sicuri di aver centrato alla nona linea la parte superiore della cornice. Ora cercate il carattere $55 e posizionatelo subito a sinistra della riga che avete creato. Cercate il carattere $49 e posizionatelo a destra della stessa linea. Ora cercate i caratteri $4A e $4B e metteteli, rispettivamente, sei linee al di sotto del $55 e del $49. Cercate il carattere $5D e fate le due linee verticali di congiunzione tra le due coppie di spigoli che avete sistemato. A questo punto, senza cercare nuovamente il simbolo di linea orizzontale per completare la figura, posizionatevi con il cursore su uno di quelli già presenti sulla window e premete il tasto G. Il carattere comparirà nel riquadro carattere. Completate la figura e scriveteci all'interno il testo di **figura 2** centrandolo con tre spazi a sinistra e tre a destra. La window che avete creato adesso può essere salvata su disco. Per fare questo dovete dgitare F7, selezionare con i tasti cursore e Return l'opzione Save Window, digitare il nome (SF> WINDOW) e premere nuovamente Return. Ora il file contenente i dati della window si trova sul disco, con il nome assegnato. Per visualizzare la window dal Basic dovete inserire nel drive il disco su cui si trova il file e digitare e lanciare il **listato 1**. Per fermare l'esecuzione del breve programma dovete premere il tasto Run/Stop. Ora che avete capito come funzionano le opzioni principali del programma non dovreste trovare di difficile comprensione il prossimo paragrafo.

Listato 1.

Routine di visualizzazione di una window di 40 x 25 caratteri situato da $4000 a $43E8.

```
10 IF F<>0 THEN 30
20 F=1:LOAD"WINDOW",8,1
30 FOR X=0 TO 999:POKE1024+X,PEEK(16384+X):NEXTX
40 GOTO 40
```

Listato 2.

Esempio di attivazione del modo multicolore e un set di caratteri ridefiniti.

```
10 POKE 53282,2:POKE 53283,7: REM COLORI 1 E 2
20 POKE 53281,0:REM COLORE FONDO
30 FOR X=55296 TO 56295:POKEX,8+6:NEXTX:REM COLORE 3
40 FOR X=0 TO 999:POKE1024+X,PEEK(16384+X):NEXTX
50 POKE 53270,PEEK(53270)OR16:REM MODO MULTICOLORE
60 POKE 53272,(PEEK(53272)AND240)+8
70 GOTO 70
```

Funzioni editor

Il programma Char Works funziona in due modi: modo carattere e modo window.

Nel primo modo il cursore lampeggiante si trova nella parte alta dello schermo, all'interno del riquadro carattere. Nel secondo modo il cursore è sulla window. In entrambi i modi, il riquadro carattere (in alto a sinistra) contiene sempre un modello ingrandito del carattere in uso. Il passaggio da un modo all'altro si effettua premendo il tasto J.

• Tasti cursore, barra spazio e asterisco

I tasti cursore sono attivi in entrambe le modalità di funzionamento. La loro funzione è quella di muovere il cursore lampeggiante nell'area a sua disposizione. Il tasto asterisco serve per scrivere un pixel nel modo carattere e un carattere nel modo screen. La barra spazio serve per cancellare un pixel (solo nel modo carattere). Per cancellare un carattere piazzato sulla window e necessario rimpiazzarlo con un carattere spazio (codice $20 in un set di caratteri standard).

Al posto dei tasti cursore è anche possibile usare un joystick posto in porta 2. Al posto della barra spazio potete invece usare il tasto fire del joystick, sempre in porta 2. Il tasto Fire del joystick in porta 1 corrisponde al tasto asterisco.

• Tasti X e Y

Questi tasti sono attivi solo in modo carattere. Il tasto X effettua un ribaltamento orizzontale del carattere nel quadro carattere. Il tasto Y effettua un ribaltamento verticale.

Tavola 1.

Conversione da esadecimale a decimale di alcuni valori per facilitare alcune funzioni del programma

0	= $00	88	= $58	176	= $b0
2	= $02	90	= $5a	178	= $b2
4	= $04	92	= $5c	180	= $b4
6	= $06	94	= $5e	182	= $b6
8	= $08	96	= $60	184	= $b8
10	= $0a	98	= $62	186	= $ba
12	= $0c	100	= $64	188	= $bc
14	= $0e	102	= $66	190	= $be
16	= $10	104	= $68	192	= $c0
18	= $12	106	= $6a	194	= $c2
20	= $14	108	= $6c	196	= $c4
22	= $16	110	= $6e	198	= $c6
24	= $18	112	= $70	200	= $c8
26	= $1a	114	= $72	202	= $ca
28	= $1c	116	= $74	204	= $cc
30	= $1e	118	= $76	206	= $ce
32	= $20	120	= $78	208	= $d0
34	= $22	122	= $7a	210	= $d2
36	= $24	124	= $7c	212	= $d4
38	= $26	126	= $7e	214	= $d6
40	= $28	128	= $80	216	= $d8
42	= $2a	130	= $82	218	= $da
44	= $2c	132	= $84	220	= $dc
46	= $2e	134	= $86	222	= $de
48	= $30	136	= $88	224	= $e0
50	= $32	138	= $8a	226	= $e2
52	= $34	140	= $8c	228	= $e4
54	= $36	142	= $8e	230	= $e6
56	= $38	144	= $90	232	= $e8
58	= $3a	146	= $92	234	= $ea
60	= $3c	148	= $94	236	= $ec
62	= $3e	150	= $96	238	= $ee
64	= $40	152	= $98	240	= $f0
66	= $42	154	= $9a	242	= $f2
68	= $44	156	= $9c	244	= $f4
70	= $46	158	= $9e	246	= $f6
72	= $48	160	= $a0	248	= $f8
74	= $4a	162	= $a2	250	= $fa
76	= $4c	164	= $a4	252	= $fc
78	= $4e	166	= $a6	254	= $fe
80	= $50	168	= $a8	1024	= $0400
82	= $52	170	= $aa	4096	= $1000
84	= $54	172	= $ac	8192	= $2000
86	= $56	174	= $ae	32768	=$8000
				49152	= $C000

• Tasti U, D, L e R

Questi tasti, attivi solo in modo carattere, permettono di scrollare il contenuto del riquadro carattere in una delle direzioni (alto, basso, sinistra e destra)

• Tasto G

Nel modo carattere il tasto G attiva un input che richiede il codice del carattere da usare. Questa

funzione velocizza notevolmente la ricerca di un carattere di cui si conosce il codice.

Nel modo window lo stesso tasto serve per visualizzare nel riquadro carattere il carattere su cui si trova il cursore, all'interno della window. Anche questa funzione velocizza la ricerca di un carattere.

• Tasti C e Shift+Inst/Del

Nel modo carattere questo tasto attiva un input che richiede il codice del carattere da copiare in overlay su quello correntemente in uso. Copia in overlay significa che la sagoma del carattere in arrivo si sovrappone a quella nel riquadro.

Per copiare normalmente un carattere al posto di quello corrente occorre cancellare quest'ultimo con i tasti Shift e Inst/Del, premuti contemporaneamente.

• Tasti addizione (+) e sottrazione (-)

In entrambi i modi di funzionamento questi due tasti servono rispettivamente per l'avanzamento e per l'arretramento del codice del carattere visualizzato nel riquadro in alto a sinistra.

• Tasti Ctrl/R

La pressione contemporanea dei tasti Ctrl e R trasforma il carattere in uso nel negativo di se stesso. Questa funzione è utilissima per creare un set di caratteri ridefiniti completi dei loro corrispettivi in reverse.

Il sistema operativo del C64, infatti, richiede, oltre alla serie di caratteri normali, la rispettiva immagine in reverse di ognuno di essi per effettuare l'effetto di lampeggiamento del cursore sovrapposto a un carattere.

• Tasto W

Il tasto W è attivo in entrambi i modi ed è usato per visualizzare a pieno schermo la window. Se la window è più grande dello schermo (vedi oltre) può essere esaminato muovendosi con i tasti cursore. Per tornare al modo corrente basta premere il tasto W.

• Tasto V

Solo modo carattere. La pressione di questo tasto visualizza l'intero set di caratteri al posto della window.

Questa opzione è molto utile per vedere a colpo d'occhio le caratteristiche del set in uso. Per tornare alla normalità occorre premere nuovamente il tasto V.

• Tasto <-

Il tasto freccia a sinistra, che è attivo solo in modo window, ha una funzione molto particolare. La funzione da esso gestita, infatti, consente di prelevare un blocco di dati dalla window e di copiarlo in un'altra zona della stessa. Questa possibilità risulta utilissima se occorre riempire un'area con elementi ripetitivi.

Se per esempio avete disegnato un albero costituito da sei caratteri disposti su due colonne di tre, lo avete posto sulla window carattere per carattere, e vi occorre riprodurlo in più punti, dovete porre il cursore sul carattere superiore sinistro del blocco da copiare, premere il tasto freccia a sinistra, muovere il cursore fino a che l'area da copiare non è tutta coperta (l'area del blocco viene coperta dai caratteri del bordo della window).

Premere nuovamente il tasto freccia a sinistra e muovere il blocco nella zona opportuna dove potete piazzarlo premendo il tasto asterisco.

Per abbandonare il blocco e tornare alla normalità occorre premere ancora il tasto freccia a sinistra.

• Tasto S

Questo tasto seleziona la zona di memoria da cui il programma trae le informazioni del set di caratteri. Le zone di memoria possibili sono due: set 1 = $0800, set 2 = $2000.

• Tasti M e H

Nel modo carattere, il tasto M attiva il modo caratteri multicolore. Il riquadro caratteri rimane suddiviso in una griglia di 4 x 8 pixel, anziché 8 x 8, ma consente di creare caratteri a quattro colori (compreso quello di fondo). Premendo il tasto H torna al modo normale.

• Tasti 1, 2 e 3

I tasti numerici dall'1 al 3 selezionano, nel modo carattere, il colore corrente.

Una freccia a sinistra degli indicatori (in alto a destra del video) indica il colore selezionato. Questa funzione risulta molto utile solo trattando caratteri multicolore.

• Tasti numerici shiftati

Shiftare i tasti significa premerli contemporaneamente al tasto Shift.

I tasti numerici shiftati permettono di cambiare i colori di display e quelli dei caratteri multicolore.

- Shift+1: cambia il primo colore dei caratteri multicolore.
- Shift+2: cambia il secondo colore.
- Shift+3 : cambia il colore numero tre, che corrisponde al colore dei caratteri non multicolore (default).
- Shift+4 : cambia il colore del bordo del riquadro carattere e degli indicatori.
- Shift+5 : cambia il colore dello sfondo.

Altre funzioni

Il tasto F7 visualizza un menù d'opzioni selezionabili con i tasti cursore e il tasto Return.

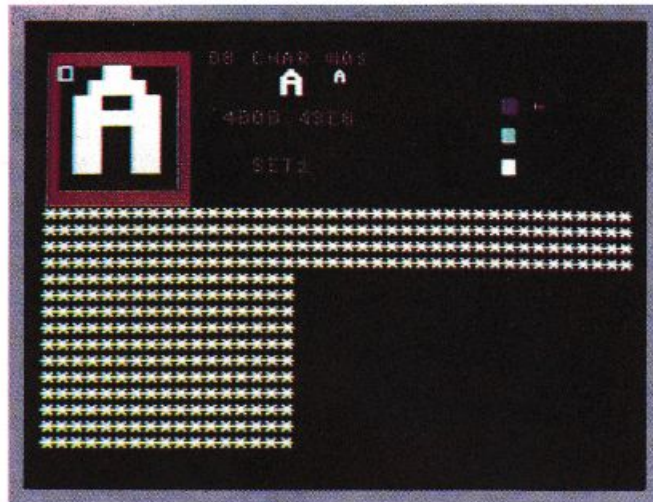



Figura 1. Schermata principale del programma

• Load

Questa funzione permette di caricare in memoria un set di caratteri o una window. È importante sottolineare che tutto ciò che viene caricato in memoria non viene riconosciuto dal programma in alcun modo.

In pratica il programma ignora se quello che caricate è una window, un set di caratteri o qualsiasi altra cosa.

Un set di caratteri viene visualizzato immediatamente solo se la zona di memoria in cui viene caricato corrisponde a quella selezionata con il tasto S, cioè solo se si trova in $0800 (set 1) oppure in $2000 (set 2).

Nel caso in cui abbiate a disposizione un set di caratteri residente in un'altra locazione dovete ricorrere all'opzione Fetch Char (vedi oltre).

• Save chars

Questa funzione salva il set di caratteri su disco. Il set salvato è, naturalmente, quello corrente (set 1 oppure set 2) e, se ricaricato, manterrà la propria ubicazione ($0800 oppure $2000).

Quando selezionate l'opzione dovete specificare il nome del file (SF>), il codice del primo carattere della parte di set da salvare e il codice del carattere finale. Per salvare tutto il set di caratteri dovete specificare come primo carattere 00 e come ultimo FF.

Un set di caratteri intero occupa in memoria, e ovviamente sul disco, mezzo Kb.

• Save window

Questa funzione, analoga alla precedente, permette di salvare la window. In questo caso è sufficiente specificare il nome del file da salvare (SF>) e premere Return. Le dimensioni del file dipendono dalle dimensioni della window che possono essere specificate mediante l'opzione Window Size (vedi oltre).

• Multi Col, Hires e View chars

L'opzione Multi Col attiva il modo multicolore e corrisponde al già menzionato tasto M. L'opzione Hires, invece, corrisponde al tasto H, anch'esso già visto sopra. Lo stesso discorso vale per l'opzione View chars che corrisponde al tasto V.

• Fetch chars

Questa opzione permette di trasferire un set di caratteri locato in una zona qualsiasi della memoria, nella memoria riservata al set in uso. Se per esempio avete un set sul disco tale che, se caricato, va a posizionarsi in $4000, per trasferirlo nel set corrente dovete caricarlo, attivare l'opzione e specificare quando richiesto (F <-), le prime due cifre esadecimali della zona in cui è locato il set.

Basatevi sulla **tavola 1** per effettuare più facilmente le conversioni dei numeri da decimali a esadecimali.

Se vengono specificati i valori $D0 oppure $D8 il set corrente assumerà le sembianze di uno dei due set standard del Commodore 64.

• Border char

Con questa opzione potete decidere il carattere che costituisce il bordo della window.

Quando richiesto (B <-) dovete semplicemente specificare il codice del carattere desiderato.

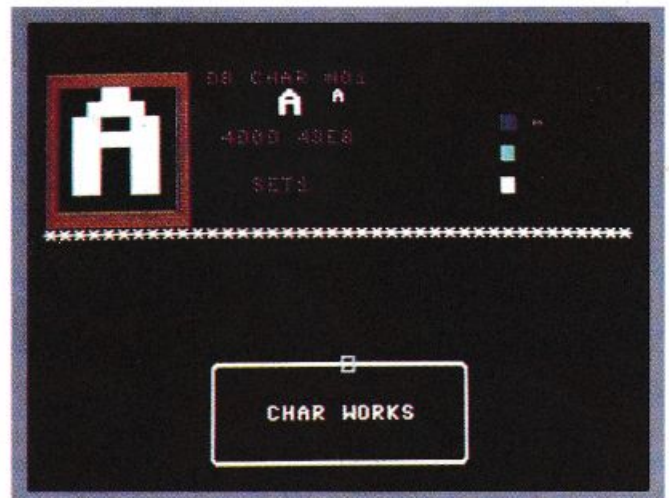



Figura 2. Lo screen costruito passo per passo nell'esempio

• **Fill window**
In modo molto analogo all'opzione precedente, questa permette di riempire la window con un carattere.

• **Window size**
Le dimensioni della window possono essere variate con questa funzione. Dovete ricordare che il programma tratta solo numeri in formato esadecimale, per cui dovete riferirvi alla **tavola 1** per effettuare una conversione quando avete deciso le dimensioni della finestra. Per effettuare il cambiamento attivate l'opzione e specificate la dimensione x e la dimensione y della window.

• **Position second**
Questa opzione permette di definire da quale dei due set di caratteri devono provenire le informazioni che danno forma ai caratteri sulla window. In pratica, quando il programma è stato appena avviato, i caratteri che vengono posti sulla window appartengono al set 1, indipendentemente da quale sia il set corrente. Ciò significa che ogni volta che volete usare i caratteri del secondo set dovete attivare l'opzione e introdurre, quando richiesto (P <-), il valore $20, che corrisponde alla posizione in memoria del set secondario. Per tornare al set primario specificate il valore $08.

• **Device**
Questa opzione consente anche agli utenti senza un drive di sfruttare le ottime funzioni di questo programma. Per sapere quale tipo di device è attivo basta guardare l'indicatore in alto, fra il quadro carattere e la parola char. La seconda cifra è otto se il device è il drive, uno se è un registratore a cassette.

• **Copy sprite**
Questa interessante opzione consente di catturare una parte di sprite caricato in memoria di cui si conosce la locazione di inizio. Quando l'opzione viene selezionata viene richiesto un valore esadecimale che corrisponde alle prime due cifre della locazione d'inizio dei dati dello sprite. Fatto questo compare in alto a sinistra lo sprite e un cursore che può essere mosso liberamente al suo interno con i tasti cursore shiftati, fino ad inquadrare la zona da copiare al posto del carattere in uso. Inutile dire che questa opzione facilita di molto la costruzione di forme costituite dall'accostamento di più caratteri.

Figura 3. Effetto bassorilievo con caratteri multicolore e un pizzico di creatività

• **Base Address**
La posizione della window può essere variata specificando le prime due cifre della locazione d'inizio della memoria da essa occupata (BH <-). Le prime due cifre, naturalmente, sono in forma esadecimale.

Modo Multicolore

Vediamo ora come attivare da Basic una window costituita da caratteri multicolore. Ipotizziamo che abbiate creato una window che va da $4000 a $43E8, grande, cioè, come lo schermo. Supponiamo che questa window sia costituita da caratteri multicolore ridefiniti e da voi salvati in $2000 (vedi tasto S) e che il set abbia i seguenti colori: il colore 1 (il primo in alto fra i tre indicatori del colore) sia rosso (codice 2, vedi manuale C64), il colore 2 sia giallo (codice 7) e il colore 3 sia blu (codice 6). Il Commodore 64 usa due locazioni per specificare i colori 1 e 2 del modo multicolore: rispettivamente la 53282 e la 53283. Il colore di fondo, invece, è definito nella locazione 53281.

La memoria colore, analoga alla memoria video da 1024 a 2023 compreso, contiene le informazioni relative al colore 3 del modo multicolore e si estende da 55296 a 56295. In particolare i primi tre bit di ognuna delle mille locazioni determinano il colore di ogni cella carattere a cui corrispondono. Per non complicare troppo le cose, noi ci limiteremo a dare un unico colore 3 a tutta la window, anche perché il programma non può gestire le window in maniera differente. Digitate il **listato 2**, salvatelo e resettate il computer. Caricate la window e il set di caratteri con LOAD"NOME",8(1),1 e il **listato 2**. Il breve programma stabilisce i colori, trasferisce i dati della window nella memoria video, aggiorna la memoria colore secondo il colore 3 e attiva, finalmente, sia il set di caratteri ridefiniti sia il modo multicolore.

Raffaele Zanini

Tips per tutti, come sempre, all'insegna dell'efficienza e della versatilità. I tips di questo mese non sono semplici utility ma potenti tool per programmatori e non. Tra l'altro c'è anche un microvideogame e un prezioso consiglio per proteggere i vostri programmi

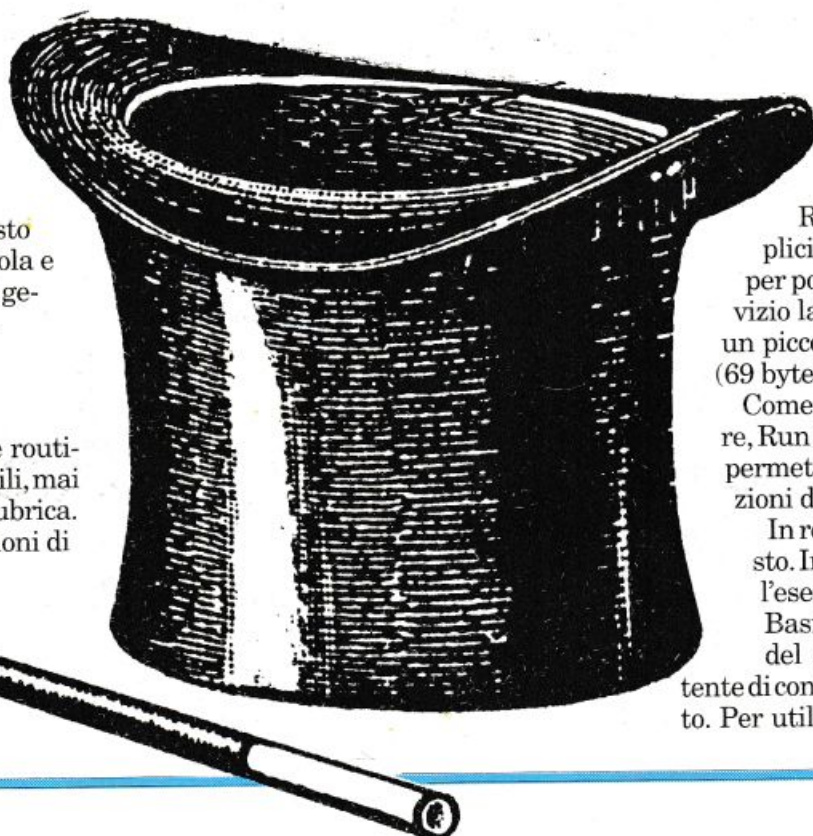
Trucchi Incredibili e Programmi Simpatici

Cominciamo la rassegna dei tips di questo numero con una piccola e utilissima routine di gestione dell'interrupt.

Run Control

Questa è una delle routine da interrupt più utili, mai apparse su questa rubrica. Le possibili applicazioni di Run Control sono molteplici e, ciò che conta di più, per poter avere al proprio servizio la routine si deve pagare un piccolissimo tributo di Ram (69 byte di codice macchina).

Come il nome lascia intendere, Run Control è una utility che permette di controllare l'esecuzioni dei programmi.

In realtà fa molto più di questo. Infatti agisce non solo sull'esecuzione dei programmi Basic ma su tutta l'attività del C64 permettendo all'utente di controllarla a suo piacimento. Per utilizzare la routine dovete

Listato 1

```
1 rem ———
2 rem -          -
3 rem - run control -
4 rem -          -
5 rem ———
6 :
10 foru=679 to 747:read q:pokeu,q:ck=ck+q:next
20 if ck<> 8389 then print"<CLEAR>errore nei data !!":end
30 print"<CLEAR>f1 - computer in pausa"
40 print"f3 - disattiva pausa finche' e' premuto"
50 print"f5 - disattiva pausa"
60 print"f7 - rallenta il computer"
70 :
80 :
90 sys 679
100 :
110 :
111 rem ———
112 rem -       -
113 rem - dati lm -
114 rem -       -
115 rem ———
116 :
120 data 120,169,180,141,20,3,169,2,141,21,3,88,96,165,197,201,3,208
130 data 10,162,10,160,255,136,208,253
140 data 202,208,248,165,2,201,1,240,6
150 data 165,197,201,4,208,13,32,159,255
160 data 165,197,201,5,208,7,169,1,133,2
170 data 76,49,234,201,6,208,236,169,0
180 data 133,2,76,49,234,0
```

copiare il **listato 1** e dare il RUN. Se non avete commesso errori di trascrizione, alla ricomparsa del ready Run Control sarà già stato attivato.

A questo punto avete a disposizione i seguenti comandi:

•**F1:** mette in pausa il computer. L'eventuale programma Basic in esecuzione verrà congelato.

Questa istruzione agisce anche, per esempio, sul list di un programma bloccandolo.

•**F3:** è possibile usarlo solo dopo aver utilizzato il programma precedente.

L'attività del computer riprenderà normalmente finché viene tenuto premuto il tasto e si interromperà nuovamente rilasciandolo.

Questo comando è utile per seguire l'esecuzione dei programmi.

•**F5:** permette di disattivare definitivamente l'effetto del primo comando, cioè fa in modo che l'attività del computer riprenda normalmente.

•**F7:** tenendo premuto questo tasto si rallenta qualsiasi attività del computer (e di conseguenza anche l'esecuzione dei programmi, del list eccetera).

Rilasciando il tasto il computer torna normale.

Motocross

È un microvideogame ancora più sofisticato, se così si può dire, di quello apparso nella puntata precedente.

Pur essendo realizzato completamente in Basic e non avendo nulla degli effetti speciali sfoggiati da ogni videogame in circolazione, Motocross è un ottimo esempio di tecnica di programmazione.

Prima ancora di giocarci provate quindi a osservare attentamente il listato e cercate di capire bene tutte le tecniche e i trucchi che sono stati adottati per ridurre al minimo e comprimere le linee necessarie e per ottenere il massimo della velocità di esecuzione (il punto (.) al posto dello zero, niente Then dopo If eccetera).

Per giocare a Motocross dovete copiare il **listato 2** e dare il RUN. Per controllare la moto dovete utilizzare i tasti M e C che vi consentono di spostarvi rispettivamente a destra e a sinistra.

Non c'è assolutamente bisogno di dare consigli di gioco per un game così semplice.

Un solo appunto: se riuscite spesso a superare i 500 punti (senza modificare il listato originale, naturalmente) sicuramente avrete un futuro come giocatori di videogame, oppure siete già dei campioni.

Listato 2

```
1 rem ———
2 rem -          -
3 rem - motocross -
4 rem -          -
5 rem ———
6 :
10 v=53248:m=54296:w=v+31:lm=220:ll=132:mc=203:t=8:l=20:r=36:c=16:z=22:p$="<HOME><DOWN 2>"
20 forj=832to894:readd:pokej,d:next:pokew+1,0:pokew+2,0:poke2040,13:pokev+1,145
30 print"<CLEAR><YELLOW><RVS ON>motocross":pokew+t,rnd(0)*6+1:poke646,peek(w+t)+3:s=-9:x=172
40 pokev,x:pokev+21,1:f=peek(w)
50 printp$spc(39)chr$(148)" "spc(c)"W       W":pokelm,ll:s=s+1:ifs<.goto50
60 printp$s:g=peek(mc):pokem,t
70 x=x+t*((g=l)-(g=r)):pokev,x:ifc<t orc=int(rnd(.)*t+c)then n=1-n
80 c=c-(n=.)+(n=1):ifc>z then c=z:n=1
90 pokem,.:on-(peek(w)=.)goto50:fore=1 to 10:pokev+39,e:getb$:next:pokev+21,0
100 printchr$(147)"<YELLOW>game over!"p$"score= "s:input"<DOWN>try again (y/n)";a$
110 if a$="y" then run
120 data ,28,,,62,,,62,,,62,,,62,,,62,,7
130 data 255,240,14,28,56,30,62,60,54
140 data 62,54,6,28,48,7,255,240,3,255,224,,255
150 data 128,,255,128,,62,,,62
160 data ,,62,,,62,,,62,,,28,,
```

Clear Without Home

Il nome fa subito capire di cosa si tratta. I pochi byte di linguaggio macchina della routine vi permetteranno finalmente di cancellare lo schermo senza alterare la posizione del cursore.

Per poter utilizzare Clear Without Home dovete copiare il **listato 3** e dare il RUN.

A questo punto, se non avete commesso errori nel trascrivere il listato (non è stato posto un checksum per il numero molto ridotto di dati da leggere), siete pronti per poter utilizzare la routine.

Ogni volta che volete cancellare lo schermo senza alterare la posizione del cursore dovete usare l'istruzione sys 49152.

Listato 3

```
1 rem ———————
2 rem -                 -
3 rem - clear without home -
4 rem -                 -
5 rem ———————
6 :
10 sa=49152:fort=0 to 19:read m
20 poke sa+t,m:next
30 data 162,0,169,32,157,0,4,157,0,5,157
40 data 0,6,157,0,7,232,208,241,96
```

No Blink Cursor

Anche in questo caso il nome della routine dice già tutto da solo. No Blink Cursor inibisce l'effetto di blinking del cursore rendendolo statico. Per chi vuole creare routine di input controllato, questa routine è assolutamente indispensabile per dare un tocco di professionalità oltre al risultato finale.

Per utilizzare la routine dovete copiare il **listato 4**, dare il RUN e quindi digitare sys 49152. No Blink Cursor si disattiva premendo contemporaneamente i tasti Run/Stop e Restore.

Listato 4

```
1 rem ———————
2 rem -                 -
3 rem - no blink cursor -
4 rem -                 -
5 rem ———————
6 :
10 fori=49152 to 49175:read d
20 poke i,d::ck=ck+d:next
30 if ck<>2660 then print"<CLEAR>errore nei data!!":end
40 sys 49152
50 data 120,169,13,141,20,3,169,192
60 data 141,21,3,88,96,165,207,240
70 data 4,169,2,133,205,76,49,234
```

Underline Cursor

Questa routine è l'alternativa alla precedente. Consente infatti di modificare l'aspetto del cursore trasformandolo nel più professionale trattino. Anche questa routine è l'ideale per dare ai vostri programmi un tocco di professionalità oppure, più semplicemente, per cambiare look all'editor. Per utilizzare Underline Cursor copiate il **listato 5** e dare il RUN. Per attivare il nuovo cursore digitate sys 828 (ciò significa che la routine può essere utilizzata solo se non usate il registratore). Per ripristinare il normale cursore dovete premere contemporaneamente i tasti Run/Stop e Restore.

Listato 5

```
1 rem ———————
2 rem -                 -
3 rem - underline cursor -
4 rem -                 -
5 rem ———————
6 :
10 fort=828 to 967:reada:poket,a:b=b+a:next
20 if b<>20972 then print"<CLEAR>errore nei data":end
30 :
40 :
50 data 169,48,133,52,133,56,133,251,173
60 data 14,220,41,254,141,14,220,165,1
70 data 41,251,133,1,169,0,133,250,133,252
80 data 133,254,169,52,133,253,169,208
90 data 133,255,32,180,3,201,212,208
95 data 249,169,56,133,251,169,60,133,253
100 data 169,216,133,255,32,180,3,201,220,208
110 data 249,169,52,133,253,160,7,169
120 data 255,145,250,145,252,24,152,105
130 data 8,168,192,7,208,241,230,251,230
140 data 253,165,253,201,56,208,231,165
150 data 1,9,4,133,1,173,14,220,9,1,141,14
160 data 220,173,24,208,41,240,9,12,141
170 data 24,208,96,160,0,177,254,145,252
180 data 145,250,200,208,247,230,251,230
190 data 253,230,255,165,255,96
```

Sequential protection

Un piccolo consiglio per proteggere i vostri programmi dagli sguardi indiscreti.

Non si tratta ovviamente di una mega protezione in grado di scoraggiare ogni tentativo di copiatura ma semplicemente di una tecnica semplicissima per impedire ai più sprovveduti di caricare facilmente i vostri programmi. Quando salvate un programma Basic mettete in fondo al nome l'estensione, S. L'estensione non comparirà se listate la directory e il programma verrà considerato di tipo sequenziale.

Ogni tentativo di caricare il programma con un normale Load sarà infruttuoso. L'unico modo di caricare il programma consiste nello specificare di nuovo l'estensione, S, nel nome del file. Ecco un esempio che chiarirà meglio il discorso:

SAVE"PIPPO,S",8

salva il programma Basic in memoria col nome Pippo. L'estensione, S, non compare nella directory ma il tipo del file è sequenziale. Per caricare il programma usate l'istruzione:

LOAD"PIPPO,S",8.

Daniele Maggio

Per non battere la testa...

Non vi siete mai preoccupati per quei sinistri rumori che provengono dal vostro drive in certe particolari condizioni di funzionamento? Avete mai dovuto spendere qualche lira di troppo e dovuto perdere tempo prezioso per far riallineare la testina del vostro drive? Se vi sentite chiamati in causa da queste domande dovete seguire questo articolo per mettervi definitivamente al riparo da certi pericoli

Come sapranno quasi tutti i possessori di questa periferica, il drive 1541, durante alcune operazioni, provoca l'urto del meccanismo di trascinamento della testina contro il fine corsa, provocando uno sgradevole rumore e il giustificato timore dell'utente per la salute del suo prezioso strumento.

Per motivi legati alla progettazione dell'hardware e del software che lo gestisce, il drive rileva la posizione della testina solo quando riesce a leggere una qualsiasi traccia di un dischetto formatta-

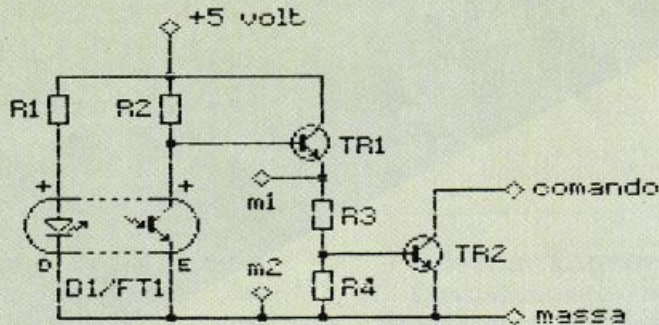



Figura 1. Schema del circuito e lista componenti

to. Questo, però, a volte non è possibile. Per esempio quando si verifica un errore di lettura o quando viene inserito un disco vergine da formattare. In questi casi, e cioè quando si verifica una situazione in cui il drive non sa con esattezza su quale traccia sia posizionata la testina, vengono inviati al motore passo-passo che controlla la testina stessa 35 comandi di spostamento verso una traccia con un numero più basso (e quindi più esterna). Da qualsiasi posizione iniziale, dopo questi comandi, la testina si trova posizionata in traccia uno, essendo presente un fine corsa meccanico che le impedisce ulteriori movimenti verso tracce ancora più esterne e contro il quale la meccanica di trascinamento ha urtato un numero imprecisato di volte e dipendente dalla posizione iniziale della testina.

Il sistema descritto sopra è semplice e funziona bene, ma presenta l'inconveniente di forzare in modo eccessivo l'accoppiamento meccanico a pressione fra l'albero del motore passo-passo e la rotella di trascinamento della testina, portante i riferimenti di fine corsa. Ciò provoca, con il tempo e con la complicità delle dilatazioni termiche, lo scorrimento relativo delle due parti e quindi la variazione della posizione della testina rispetto alle tracce del dischetto, con conseguenti errori di lettura e rendendo spesso necessarie spese e perdite di tempo per il riallineamento del drive.

Il circuito proposto è stato studiato per eliminare gli urti contro il fine corsa, mediante il parziale blocco dei comandi per il motore passo-passo, quando la testina si trova sulla traccia uno e i comandi stessi cercano di portarla su tracce ancora più esterne.

Il blocco completo dei comandi al motore avrebbe reso necessario un circuito di decodifica piuttosto complicato e si è quindi optato per una soluzione che, pur non bloccando del tutto i comandi stessi, diminuisce moltissimo la forza applicata dal motore nella situazione prima descritta ed è nel contempo economica e di semplice installazione.

Per riconoscere quando la testina è sulla traccia uno vengono usati un diodo emettitore di luce e un fototransistor, disposti in modo che la luce emessa dal diodo e ricevuta dal fototransistor sia intercettata da una aletta presente a lato del carrello portatestina. I risultati ottenuti, purché la parte ottica sia messa a punto, sono ottimi: rumore quasi del tutto scomparso, compatibilità con tutti i programmi, anche protetti, e con le varie versioni di Speeddos e cartucce velocizzatrici, nessuna alterazione della taratura originale.

Descrizione del circuito

Il funzionamento del circuito, il cui schema è riportato in **figura 1,** è il seguente: se la testina si trova su una traccia diversa dalla uno, la luce emessa dal diodo D1, facente parte del Cny36, cade sul fototransistor Ft1, anche esso parte del Cny36, e lo mantiene saturato.

La tensione di base di Tr1 è di circa 0,3 volt e la sua tensione di emettitore è circa zero volt, così che Tr2 è interdetto e il suo collettore, collegato alla base di Q11 sulla piastra del drive, può variare liberamente di tensione, consentendo il normale funzionamento.

Quando la testina arriva sulla traccia uno, la luce incidente su Ft1 viene bloccata ed esso si interdice quasi completamente, la sua tensione di collettore sale a circa 3,5 volt e la tensione di emettitore di Tr1 si porta a circa tre volt saturando Tr2, il quale mantiene a circa 0,1 volt la base

di Q11, bloccando il segnale indicato con la lettera "d" in **figura 2** ed evitando che il motore passo-passo applichi la sua piena forza nel tentare di spostare la testina. I movimenti di quest'ultima verso la traccia 2 non vengono bloccati in quanto il comando 'd' è attivo soltanto durante il passaggio fra le tracce 2-3, 4-5 eccetera.

Realizzazione pratica

Sulla casetta trovate un breve programma che si chiama Testn e che servirà per testare il lavoro fatto. Spegnete calcolatore e drive e staccate da questo le spine del cavo seriale e di alimentazione, aprendolo ed estraendo la meccanica. Preparate la parte ottica incollandovi due striscette di sottile lamierino o due fili con una goccia di cianoacrilato e montatela sulla meccanica. Riferitevi alla **figura 3** per sapere come il tutto, fissato con una piccola vite, debba essere posizionato rispetto al carrello portatestina. Le **figure 3** e **4** sono utili come guida per la costruzione e il fissaggio del circuito, tenendo presente che quello raffigurato è un prototipo montato piuttosto velocemente e con poca cura, per cui si possono trovare soluzioni migliori, non essendovi alcun elemento critico. Si raccomanda comunque di eseguire il collegamento della parte ottica al circuito servendosi di un piccolo connettore, per rendere più semplici eventuali smontaggi successivi del drive.

I tre collegamenti necessari fra il circuito e la piastra elettronica del drive dovranno essere realizzati con fili sottili e con una certa attenzione, per non danneggiare i componenti originali o le piste del circuito stampato. I punti di collegamento sono i seguenti:

1) +5 volt dal terminale di L2 rivolto verso l'alto, vicino al quarzo.

2) Massa dal fissaggio della basetta o da una delle piazzole vuote vicino al quarzo.

3) Comando dal terminale di R50 rivolto verso il transistor Q11 (questo collegamento non deve essere effettuato, per il momento).

Terminata la fase di installazione e dopo aver controllato di non aver fatto errori, ricollegate elettricamente la meccanica alla piastra elettronica senza, però, inserirla completamente sotto quest'ultima.

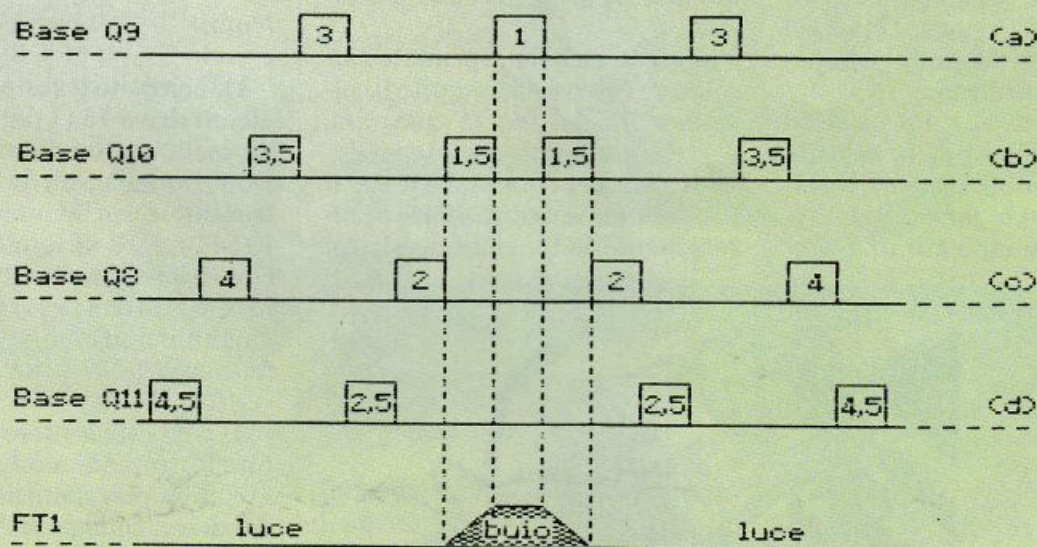



Figura 2. Relazione temporale fra gli impulsi di comando del motore passo-passo e la luce incidente su Ft1. I numeri nei rettangoli indicano su quale traccia si trova la testina in quell'istante

Allineamento

Collegate un tester con portata dieci volt c.c. fra i punti 'm1' (+) ed 'm2' (-) dello schema, inserire il cavo seriale e la spina di alimentazione del drive e accendere questo e il calcolatore, quindi caricare da nastro o da disco il programma sopracitato. Lanciate il programma ed eseguite quanto inizialmente vi sarà chiesto. Apparirà il menù di selezione lavoro, dal quale potrete comandare la posizione della testina con F1, F3 e i tasti da uno a nove. L'attuale posizione è indicata sulla riga inferiore dello schermo.

Usando i tasti F1 e F3 spostate la testina, osservando la posizione indicata sullo schermo e tenendo d'occhio la lancetta del tester, che deve indicare una tensione inferiore a 0,2 volt sulle tracce da due a 35, portandosi a oltre tre volt sulla traccia uno. Se ciò non accade, ritoccate con cautela la posizione della parte ottica e riprovate. Ottenute le tensioni suindicate, controllate in particolare che il passaggio della testina dalla traccia due alla tre e successivamente alla quattro non provochi nessun aumento, neppure transitorio, della tensione, ma soltanto la sua diminuzione oppure nessuna variazione .

Se tutto va bene, spegnete calcolatore e drive ed eseguite il collegamento del filo di controllo, precedentemente lasciato scollegato. Riaccendete, ricaricate il programma e ricontrollate accuratamente le tensioni descritte, usando anche il tasto F5 per far battere la testina contro il fine corsa (notate la differenza?). Per terminare, fissate la parte ottica sullo chassis con un po' di colla, per evitare che le vibrazioni la spostino. Il buon funzionamento del circuito dipende dalla precisione con cui viene messa a punto la parte ottica. Se questa non è perfettamente posizionata, il drive modificato non funzionerà correttamente. Il lavoro è così concluso, e non rimane altro da fare che spegnere tutto e rimontare il drive, prestando attenzione a non spostare la parte ottica durante il rimontaggio. Il vostro apparecchio è pronto per essere usato, più silenzioso e soprattutto più affidabile di prima.

Figura 3. Montaggio della piastrina con i componenti elettronici a fianco della piastra del drive. Il fissaggio è assicurato da due saldature, che portano anche il collegamento di massa del circuito. È visibile anche il connettore di collegamento della parte ottica, situata in basso a destra della foto, e i cui terminali sono identificati dalle lettere D,M,E

Figura 4. Particolare della parte ottica montata sul drive e dell'aletta presente sul carrello porta testina

Note:

1) Il circuito è stato montato su alcuni drive 1541 del tipo a scheda corta con meccanica Mitsumi D500, quella con la levetta sul frontale, e con buoni risultati. Per i 1541 a scheda lunga, dotati anche di meccaniche diverse, sarà necessario trovare i punti di collegamento opportuni e cambiare il fissaggio della parte ottica.

2) Chi desiderasse inserire un interruttore di esclusione della modifica può montarlo sul retro del drive, collegandolo al circuito in modo che, quando se ne desidera l'esclusione, venga portata a massa la base di Tr2 oppure venga interrotta l'alimentazione a cinque volt del circuito stesso.

Piero Pratesi

Tutto, facilmente!

Special Basic è estremamente potente anche nella gestione della grafica. Ne avrete una prova inequivocabile in questa puntata dedicata alle istruzioni per la manipolazione dei caratteri programmabili e degli sprite

In questa puntata vedremo tutte le istruzioni messe a disposizione dall'espansione per manipolare i caratteri programmabili e gli sprite.

Input controllato

• **Fetch:** con questa istruzione si possono risparmiare decine e decine di linee Basic. Fetch implementa infatti una potente routine di input controllato. Prima di proseguire con le spiegazioni provate, a lanciare questo programma:

```
5 cls
10 fetch "13579ad+",2,a$
20 print a$
```

Quando compare il cursore statico provate a digitare qualcosa. Non vi è possibile inserire più di due caratteri ed entrambi dovranno appartenere alla stringa che compare accanto a fetch. Inoltre, durante l'immissione dei caratteri, potete usare il tasto Delete per correggere eventuali errori. La sintassi è: fetch c$, l, s$. La stringa c$ contiene i caratteri accettati in input dall'istruzione. Il parametro l rappresenta il numero massimo di caratteri che si possono inserire e s$ è la stringa in cui sarà copiata la sequenza di caratteri digitata.

• **Getkey:** anche in questo caso un esempio chiarirà immediatamente il funzionamento dell'istruzione.

```
10 print "Un tasto"
20 getkey t$:print t$
```

La linea 20 è equivalente alle seguenti linee del Basic standard:

```
19 t$=""
20 get t$:if t$="" then 20
30 print t$
```

Figura 1. Demo delle istruzioni per la manipolazione dei caratteri programmabili

Gestione del registratore

• **Merge:** carica da cassetta un programma e lo unisce a quello, eventualmente, presente in memoria. La sintassi è: merge "nomefile" dove nomefile rappresenta ovviamente il nome del file da caricare.

• **Héader:** visualizza l'header di un programma. Sintassi come quella dell'istruzione precedente.

- *<-l:* carica un programma in modo turbo.
- *<-s:* salva un programma in modo turbo.
- *<-v:* verifica un programma salvato in modo turbo.
- *<-r:* carica un programma in modo turbo e lo lancia.
- *<-m:* esegue il merge in modo turbo (questa istruzione è quindi identica a merge, ma per il caricamento viene usato il turbo tape).

Le ultime cinque istruzioni hanno la stessa sintassi dell'istruzione merge.

Gestione della stampante

• **Type:** si comporta come l'istruzione Print (e ha la stessa sintassi di questa istruzione) con la sola differenza che i parametri passati vengono stampati e non visualizzati. Ecco un esempio:

type a$,"pippo",chr$(13);a

stampa, in ordine, il contenuto della variabile stringa a$, la stringa pippo, un Return e il contenuto della variabile a.

• **Textcopy:** effettua l'hardcopy dello schermo in modo testo.

• **Graphcopy:** effettua l'hardcopy della pagina grafica. L'istruzione ha successo se è collegata e accesa una delle seguenti stampanti: mps 801, mps 803 e compatibili.

Caratteri programmabili

• **Nchar:** copia in Ram l'immagine dei caratteri (il Vic prenderà ancora le informazioni dalla Rom per visualizzare i caratteri). Questa istruzione deve essere all'inizio del blocco di programma che ridefinisce il set di caratteri. L'immagine dei caratteri viene caricata a partire dalla locazione $E000.

• **Switch:** obbliga il Vic a servirsi dei dati in Ram per visualizzare i caratteri. Questa istruzione serve quindi per abilitare un nuovo set di caratteri. La sintassi dell'istruzione è: switch n dove n può assumere i seguenti valori:

- n=1: attiva il nuovo set. Quando il nuovo set di caratteri è attivato la memoria video è posta in $CC00 (52224) e i blocchi per gli sprite partono da $C000;
- n=0: attiva il set standard. In questo caso la memoria video è, come al solito, in $0400 e i blocchi sprite partono da $0000.

• **Chrinv:** inverte un carattere cioè trasforma un carattere nel suo Reverse (l'effetto è visibile solo se è stato attivato il nuovo set di caratteri). La sintassi è: chrinv c, s. Il primo parametro è il codice video del carattere che si vuole mettere in Reverse mentre s è il set al quale il carattere appartiene:

- s=0: set maiuscolo.
- s=1: set minuscolo.

• **Twist:** ruota un carattere. La sintassi è: twist c, s, a. I primi due parametri hanno lo stesso significato degli omonimi parametri dell'istruzione precedente mentre a rappresenta l'ampiezza della rotazione in multipli di 90 gradi in senso antiorario. Il **listato 1** contiene un esempio che dovrebbe chiarire una volta per tutte come funzionano queste ultime istruzioni.

• **Mirx:** inverte specularmente l'immagine di un carattere rispetto all'asse x. La sintassi è identica a quella dell'istruzione Chrinv.

• **Miry:** inverte specularmente l'immagine di un carattere rispetto all'asse y. La sintassi è identica a quella dell'istruzione Chrinv.

• **Create:** permette di definire un carattere. La sintassi è: create 0, c, s. Il primo parametro deve sempre essere zero (0). I due parametri successivi hanno lo stesso significato degli omonimi parametri dell'istruzione Chrinv. Questa istruzione serve per selezionare il carattere da ridefinire. La nuova definizione viene specificata per mezzo dell'istruzione Char, descritta di seguito.

• **Char:** permette di definire una linea del carattere selezionato con

l'istruzione precedente. Piuttosto che descrivere a parole come funziona utilizzeremo un esempio:

```
10     cls:nchar:switch:print"AAAAAAAAAA"
15 getkey a$
20 create 0,1,0
30 char"....****"
40 char"....****"
50 char"....****"
60 char"....****"
70 char"****...."
80 char"****...."
90 char"****...."
100 char"****...."
110 getkey a$:switch 0
```

La stringa posta dopo ogni istruzione Char definisce una linea del carattere. I caratteri che si possono usare in questa stringa sono due:

•: rappresenta un punto trasparente
*: rappresenta un punto

Volendo si può anche specificare quale linea definire con una istruzione Char. Ecco un esempio:

```
10 nchar:switch 1
20 create 0,1,0
30 char "********",1
40 char "********",3
50 char "********",5
60 char "********",7
```

Questo programma modifica le linee dispari della lettera A.

Il numero che compare dopo ogni stringa indica la linea del carattere che deve essere definita. La prima linea è identificata dal numero uno.

• **Chrcopy:** copia la definizione di un carattere in quella di un altro carattere. La sintassi dell'istruzione è: chrcopy c, s, ts, cd, sd. I primi due parametri rappresentano rispettivamente il codice video del carattere di cui si vuole copiare l'immagine e il suo set di appartenenza (maiuscolo o minuscolo). Il parametro ts permette di specificare se si deve prelevare l'immagine dal set standard, ts=0, o da quello ridefinito, ts=1. Gli ultimi due parametri sono analoghi ai primi due e individuano il carattere destinazione. Ecco un esempio:

```
nchar:switch 1:chrcopy 1,0,1,2,0
```

queste istruzioni sostituiscono all'immagine del carattere B l'immagine del carattere A del set ridefinito.

• **Chror:** esegue l'Or fra le immagini di due caratteri e deposita il risultato nell'immagine di un terzo carattere. La sintassi è: chror c1, s1, ts1, c2, s2, ts2, c, s. I primi sei parametri individuano le due immagini con cui si deve effettuare l'Or. Gli ultimi due parametri rappresentano l'immagine destinazione. Ecco un esempio:

```
10 nchar:switch 1
20 chror 1,0,1,2,0,1,32,0
```

Questo programma esegue l'Or fra le immagini della A e della B e assegna il risultato al carattere spazio.

• **Chrand:** è analoga alla precedente e ha la stessa sintassi, con la sola differenza che esegue l'And invece dell'Or.

• **Scroll:** scrolla l'immagine di un carattere. Vediamo subito un esempio per avere un'idea della potenza di questa istruzione:

```
10 switch 1:nchar
20 fchar 0,0,40,25,0
30 repeat
40 scroll d 0,0,1,1
50 get a$
60 until a$="x"
```

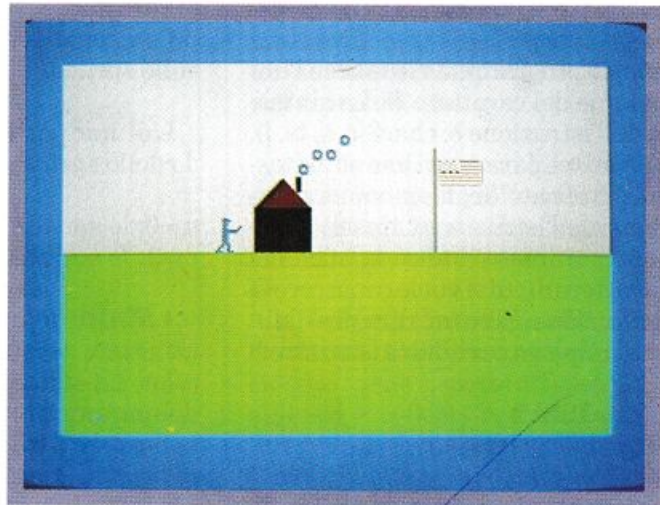



Figura 2.
Demo delle istruzioni per la manipolazione degli sprite

Con questa istruzione potete scrollare l'immagine di un carattere in uno dei quattro versi. Il verso dello scrolling è individuato dal primo parametro:

d=scrolling verso il basso
u=scrolling verso l'alto
r=scrolling verso destra
l=scrolling verso sinistra

Lo scrolling può essere con o senza wrap around e tale modalità è controllata dall'ultimo parametro:

1=scrolling con wrap around
0=scrolling senza wrap around

• **Empty:** cancella l'immagine di un carattere. La sintassi è identica a quella dell'istruzione Chrinv.

• **Char$:** è un'utilissima funzione che permette di conoscere il modo in cui è definito un carattere. Questa funzione ritorna una stringa, nel formato che si usa con l'istruzione Char, che rappresenta una linea del carattere. Ecco un esempio:

```
10 for i= 1 to 8: print
```

char$(0,0,0,i):next

questo programma visualizza l'immagine del carattere @. La sintassi dell'istruzione è: char$(c, s, ts, l). I primi tre parametri hanno lo stesso significato degli omonimi parametri dell'istruzione Chrcopy. L'ultimo parametro indica la linea del carattere di cui si vuole conoscere la definizione. Ecco un'altra possibilità di impiego per questa istruzione:

create 0,10,1
char char$ (10,1,1,2),1

queste due istruzioni cambiano la prima linea del carattere di codice video 10 con il suo Reverse.

Sprite

Le istruzioni di questo gruppo sono estremamente potenti e semplificheranno notevolmente la gestione degli sprite. Vediamole:

• **Defmob:** definisce tutti gli attributi di uno sprite. La sua sintassi è: defmob n, b, c, p, m. Il primo parametro rappresenta il numero dello sprite e deve essere compreso fra uno e otto. Il parametro b indica il numero del blocco in cui risiede la definizione dello sprite. Questo parametro deve essere compreso fra zero e 255. Tenete presente che se non è stato attivato un nuovo set di caratteri l'area di memoria utilizzabile per memorizzare gli sprite parte da $0000. Se invece è stato attivato un nuovo set la zona di memoria utilizzabile parte da $C000 e termina in $CA00. (Questa zona naturalmente può anche essere usata per routine in lm o per qualsiasi altro genere di dati). Il parametro c rappresenta il colore individuale dello sprite (è individuale perché può essere scelto liberamente per ogni sprite). Il penultimo parametro rappresenta la priorità dello sprite:

p=0: lo sprite ha la priorità rispetto ai caratteri
p=1: i caratteri hanno la priorità sullo sprite

L'ultimo parametro indica il modo dello sprite:

m=0: lo sprite è monocromatico
m=1: lo sprite è multicolor

• **Multi:** definisce i due colori da utilizzare per tutti gli sprite multicolor. La sintassi è: multi c2, c3. I due parametri sono rispettivamente il colore 2 e 3 degli sprite multicolor (cioè questi colori sono comuni a tutti gli sprite multicolor). Il colore definito nell'istruzione precedente è chiamato colore 1.

• **Mobex:** espande uno sprite in una delle due direzioni. La sintassi è: mobex n, x, y. Ecco il significato dei parametri:

n: è il numero dello sprite
x=0: dimensione normale in direzione x
x=1: dimensione doppia in direzione x
y=0: dimensione normale in direzione y
y=1: dimensione doppia in direzione y

• **Clear n:** cancella uno sprite, cioè pone a zero tutti i byte che ne definiscono l'immagine. La sintassi è: clear n, dove n rappresenta il numero dello sprite da cancellare.

• **Mobset:** fissa la posizione di uno sprite. La sintassi è: mobset n, x, y. Il primo parametro rappresenta il numero dello sprite. Gli altri due parametri sono, rispettivamente, l'ascissa e l'ordinata del punto in cui si vuole spostare lo sprite. Per l'ascissa potete usare valori compresi fra zero e 511 (niente più msb, per fortuna) mentre per l'ordinata valori nel range zero-255.

• **Mmob:** muove da interruzione, quindi automaticamente, uno sprite da una posizione a un'altra. La sintassi è: mmob n, x1, y1, x2, y2, v. Il primo parametro rappresenta il numero dello sprite. I quattro parametri successivi rappresentano le coordinate del punto di partenza e d'arrivo dello spostamento. L'ultimo parametro è la velocità dello sprite:

v=0: velocità massima
v=255: velocità minima

• **Move:** muove da interruzione uno sprite dalla sua posizione corrente a un'altra. La differenza rispetto all'istruzione precedente sta nel fatto che con questa non è necessario indicare la posizione di partenza. Ecco la sintassi: move n, x, y, v. I parametri n e v hanno lo stesso significato degli omonimi parametri dell'istruzione precedente. I parametri x e y sono le coordinate del punto d'arrivo.

• **Clrvic:** inizializza il Vic, il circuito integrato che gestisce la grafica. Questa istruzione, che non necessita di alcun parametro, dovrebbe sempre essere posta all'inizio del programma.

• **Bump:** permette di testare le collisioni fra due sprite e fra uno sprite e un carattere. La sintassi di questa istruzione è duplice:

- bump (n): verifica se lo sprite n è entrato in collisione con un carattere
- bump (n1, n2): verifica se gli sprite n1 e n2 sono entrati in collisione

In entrambi i casi la funzione ritorna il valore uno se c'è stata la collisione e zero in caso contrario.

• **Create:** permette di definire uno sprite.

La sintassi è: create 1, b, d. Il primo parametro deve sempre essere 1 (serve per distinguere questa istruzione da quella che permette di definire un carattere). Il secondo para-

Listato 1.

Esempio di applicazione delle istruzioni Nchar, Switch, Chrinv e Twist. Il cls alla linea 10 ha effetto sulla memoria video in $CC00 perché viene eseguito dopo Switch

```
10 nchar
20 switch 1
30 cls
40 fort=0 to 12
50 print"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"chr$(157)chr$(29);
60 next
70 print:print"un tasto"
80 getkey a$
90 for y=0 to 3:chrinv 1,0:pause 1:next
100 print:print"un tasto"
110 getkey a$
120 for y=0 to 3:twist 1,0,1:pause 1:next
130 print:print"un tasto"
140 getkey a$:switch 0
```

metro rappresenta il blocco in cui si vuole porre la definizione dello sprite. Il terzo parametro indica al Vic dove sono posti all'interno della memoria i blocchi:

- d=0: i blocchi sono memorizzati a partire dalla locazione $0000 (quindi il blocco 0 inizia alla locazione 0, il blocco 1 alla locazione 64, e così via);
- d=1: i blocchi sono memorizzati a partire dall'indirizzo $4000 (quindi il blocco 0 inizia alla locazione 16384);
- d=2: i blocchi sono memorizzati a partire dall'indirizzo $C000.

• **Mob:** deve sempre essere posta dopo l'istruzione precedente e permette di definire uno sprite.

In pratica l'istruzione Create 1 seleziona il blocco in cui mettere la definizione dello sprite e questa istruzione lo riempie.

Questa istruzione funziona in modo analogo all'istruzione Char. Ecco un esempio:

```
10 create 1, 13, 0
20 mob dup ("*", 24)
30 for i=1 to 19
40 mob "*" + dup (".",22)+"*"
50 next
60 mob dup ("*",24)
70 defmob 1, 13, 0, 0, 0
80 mobex 1, 0, 0
90 mobset 1, 100, 100
```

• **Turnmob:** ruota di 90 gradi la definizione di uno sprite. La sintassi è turnmob b, d, a.

I primi due parametri hanno lo stesso significato degli omonimi parametri dell'istruzione Create. Il parametro a indica l'ampiezza della rotazione in multipli di 90 gradi.

• **Blcopy:** copia un blocco sprite in un altro blocco. La sintassi è: blcopy b1, d1, b2, d2. I primi due parametri individuano il blocco da copiare e gli ultimi due quello destinazione.

• **Blor:** è analoga all'istruzione precedente con la sola differenza che viene eseguito un Or orientato ai bit fra i due blocchi. La sintassi è: blor b1, d1, b2, d2, b, d.

I primi quattro parametri rappresentano i due blocchi tra cui viene eseguito l'Or e gli ultimi due parametri rappresentano il blocco in cui sarà depositato il risultato.

• **Bland:** è analoga all'istruzione precedente con la sola differenza che in questo caso tra i due blocchi viene eseguito un And orientato ai bit. La sintassi è identica a quella dell'istruzione precedente.

• **Erase:** pone a zero tutti i byte di un blocco sprite. La sintassi è: erase b, d. Il significato dei parametri è quello solito.

• **Blinv:** cambia l'immagine di uno sprite nel suo Reverse. La sintassi è identica a quella dell'istruzione precedente.

• **Reflectx:** effettua l'inversione speculare rispetto all'asse x di uno sprite. La sintassi è identica a quella dell'istruzione Erase.

• **Reflecty:** è analoga alla precedente con la differenza che l'inversione avviene rispetto all'asse y. La sintassi è: reflecty b, d, m. I primi due parametri hanno il solito significato mentre m rappresenta il modo dello sprite:

- m=0: sprite monocromatico
- m=1: sprite multicolor

• **Scrmob:** scrolla l'immagine di uno sprite. La sintassi è: scrmob dir, b, d, a, m. Il primo parametro indica il verso dello scrolling:

dir=u: scrolling verso l'alto
dir=d: scrolling verso il basso
dir=r: scrolling verso destra
dir=l: scrolling verso sinistra

I due parametri successivi hanno il solito significato. Il parametro a rappresenta il numero di linee da scrollare. L'ultimo parametro ha il seguente significato:

m=0: scrolling senza wrap around
m=1: scrolling con wrap around

• **Sprite$:** permette di conoscere il modo in cui è definita una linea del blocco sprite. La sintassi è: sprite$(b, d, l, m). I primi due parametri hanno il solito significato. Il parametro l indica la linea di cui si vuole vedere la definizione e m è il modo, multicolor o monocromatico, dello sprite.

Nelle **figure 2** e **3** sono riportati i listati di due demo che mostrano degli esempi di applicazione di tutte le istruzioni viste sin quì.

Daniele Maggio

Non chiudete quella finestra!

Sul numero precedente abbiamo introdotto il programma Finestra Hardware cercando di spiegarne i principi di funzionamento. Ora ne approfondiamo la conoscenza descrivendo tutte le routine che lo costituiscono

Il programma Finestra Hardware è un insieme di routine che permettono di gestire la finestra in tutti i modi possibili e, fornisce tutta una serie di comandi grafici per disegnarci dentro.

Descrizione dei comandi

Per mezzo di un'interfaccia Basic tutte le routine sono facilmente accessibili, anche se i risultati migliori si hanno in linguaggio macchina.

La procedura in Basic è la seguente :

sys 49152,n° routine : PAR1, PAR-2,...)

I Par indicano una serie di parametri utili alla routine. Le routine sono 15 in tutto (da zero a 14). Vediamole:

• Routine Zero

Questa routine richiede cinque parametri, tutti rigorosamente separati da una virgola:

sys 49152,0:PAR1,PAR2,PAR-3,PAR4,PAR5

È bene osservare che dopo il numero di routine vanno messi i due punti (:). Il primo parametro stabilisce il colore dei punti della finestra. Il secondo e il terzo controllano l'espansione lungo l'asse x e y rispettivamente. (Par = 0 non espanso. Par =1 espanso) Più in generale il valore di Par2 indica la distanza tra gli sprite di una stessa riga, mentre Par3 la distanza tra le varie righe. Valori di Par2 minori di 24 provocano sovrapposizioni degli sprite affiancati, maggiori di 24 distanziamenti (lo stesso vale per Par3; vedi demo 2). Il quarto parametro è la priorità (Par4 = 0 la finestra sarà nel piano sopra rispetto ai caratteri. Par4 = 1 la finestra si posizionerà sotto rispetto ai caratteri). Par5 attiva o meno il modo multicolor (Par5 = 0 modo alta risoluzione. Par5 = 1 modo multicolor inserito). I colori 2e 3 nel modo multicolor vanno pokati nelle locazioni $D025 e $D026 (53322-53323). Vediamo un esempio:

SYS 49152,0:7,0,1,0,0

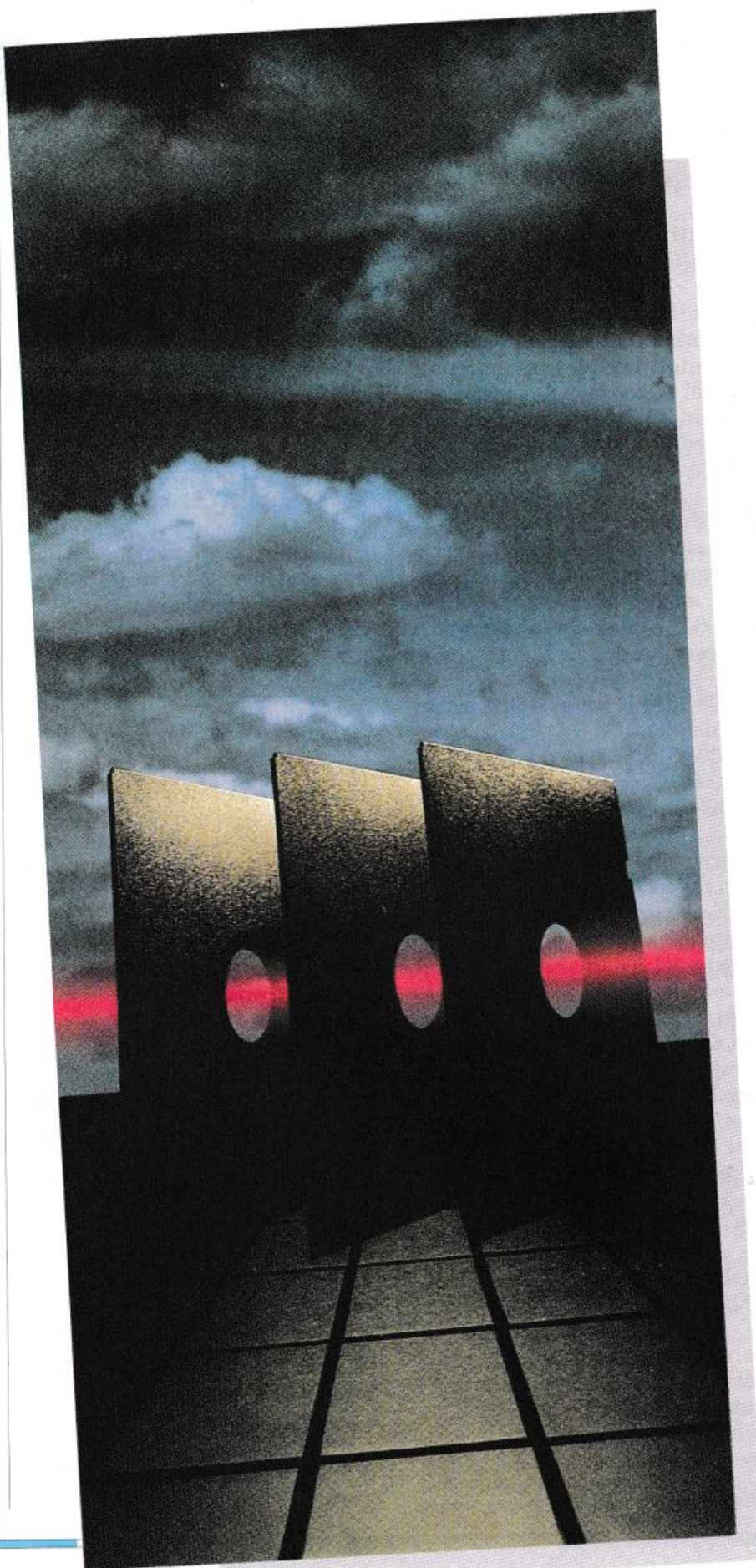
Finestra gialla (colore 7), non espansa in X ma solo in Y, posta sopra lo schermo, non in modo multicolor. Si osservi che la routine Zero modifica le caratteristiche degli sprite che compongono la finestra. Queste modifiche saranno evidenti quando la finestra sarà attivata con la routine Uno, oppure se gli sprite sono già presenti sullo schermo. Inoltre, per ottimizzare i sincronismi, cioè rendere meno brusche le modifiche, queste verranno eseguite quando il pennello elettronico sta disegnando la parte inferiore del bordo (riga dalla 250 in poi).

• Routine Uno

sys 49152,1:PAR1,PAR2,PAR3

Questa routine attiva gli interrupt raster e dunque rende fisicamente presente la finestra. Il primo parametro, compreso tra zero e cinque, indica il numero di sprite appaiati, praticamente la larghezza della finestra. Par1=0 corrisponde a un solo sprite, mentre Par1=5 corrisponde a sei sprite. Par2, che varia tra zero e 11, indica il numero di sprite in verticale (altezza della finestra). Il terzo parametro, che può al massimo valere 255, è il blocco di memoria dove si trovano i dati della finestra. Vale la formula:

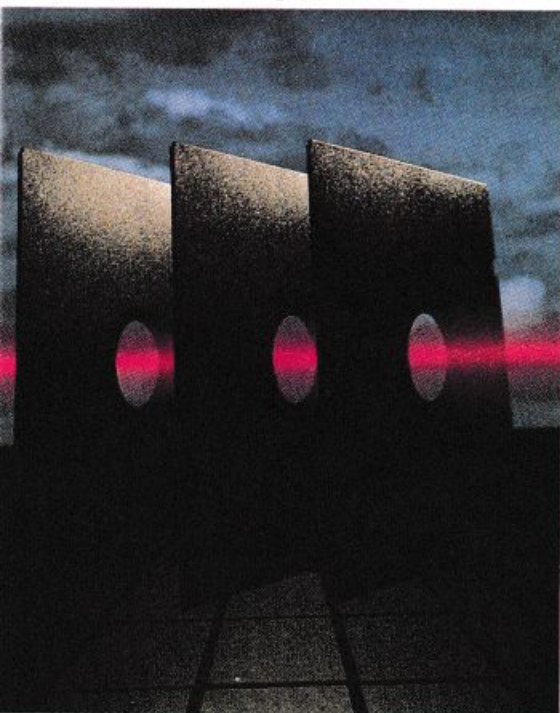
Memoria Finestra = blocco*64 + banco*16384

per blocco e banco vedi anche la routine tre. Bisogna fare molta attenzione nella scelta del blocco, perché l'area di memoria determinata dal valore del blocco sarà usata dalla finestra. Si supponga che la finestra sia formata da sei sprite, e che il

blocco scelto sia 253. I blocchi, e di conseguenza, la memoria usata dalla finestra è la seguente : blocco 253 (locazioni da 253 x 64 fino a 253 x 64+63) per il primo sprite, blocco 254 per il secondo, 255, 0, 1 e blocco 2 per l'ultimo sprite, cioè i dati dell'ultimo si trovano a 64 x 2=128. C'è allora il rischio di scrivere nelle locazioni della pagina zero, o in generale in locazioni pericolose. Un esempio:

sys 49152,1:3,5,130

La finestra ha quattro sprite di larghezza, sei di altezza e occupa la memoria a partire da 130 x 64+banco x 16384 =8320 (normalmente banco vale $0000). Notare che la routine Uno assegna alla finestra un'area di memoria contigua, ma la memoria occupata dalla finestra può anche non esserlo, cioè può trovarsi in vari punti dei 16 Kb visti dal VicII. Esiste infatti una tabella (Tabbl) allocata a $CD8B (52619 decimale) che contiene i blocchi di inizio di ogni riga di sprite. Nella tabella si trovano i blocchi del primo sprite di ogni riga, mentre gli altri blocchi della stessa riga sono ottenuti sommando uno al blocco di partenza. Se si pokano in questa tabella i valori desiderati si può sistemare la memoria occupata come meglio si vuole. Se si vogliono modificare i valori della Tabbl è ugualmente necessario aver già chiamato la routine Uno, perché questa si occupa di far eseguire tutta una serie di preparativi indispensabili:

SETX($C01B) SETY($C085) SETSP ($C16B) SETBL ($C0E9) SETINT ($C0FF)

La routine che genera la Tabbl è allocata a ($C0E9) e si chiama Setbl. È logico che l'area Ram dedicata alla finestra non può sovrapporsi ai dati di un programma Basic. La cosa migliore, per chi non è molto esperto di puntatori e aree di memoria, è di settare il banco 2 (vedi

routine Tre), e abbassare la cima della memoria Basic :

POKE 56,128:CLR

Se facciamo subito un po' di conti ci accorgeremo della dimensione massima che può assumere la finestra. 6 x 24=144 pixel in orizzontale e 12 x 21=252 pixel in verticale. Cioè lungo l'asse y la finestra ha dimensioni maggiori dello schermo visibile. Se la finestra con le dimensioni massime è posta sullo schermo, una sua parte non sarà visibile, tutti i comandi di disegno funzionano anche sulla parte non visibile. Variando la posizione y (ordinata) con la routine Due, si possono ottenere effeti di scroll fine (smooth scrolling).

• Routine Due
sys 49152,2:PAR1,PAR2

Par1, compreso tra zero e 390, rappresenta la posizione X dell'angolo in alto a sinistra della finestra. Par2, compreso tra zero e 255, è la posizione Y della finestra. Va ricordato che in certe posizioni la finestra funziona meglio, e che l'area visibile dello schermo è compresa tra i seguenti valori:

23<X<344 49<Y<250

• Routine Tre
sys 49152,3:PAR1,PAR2,PAR3

Il Vic II può vedere solo 16 Kb di memoria contemporaneamente, pertanto è necessario sapere quale area di memoria è a disposizione del Vic II. Le quattro partizioni possibili dei 64 Kb sono chiamate banchi. In questi 16 Kb devono esserci i dati degli sprite (e quindi della finestra), 1 Kb Ram per la memoria video e due Kb per il generatore caratteri. Il banco si può spostare all'interno dei 64 Kb di Ram del C64, in modo che la memoria a disposizione del Vic II sia :

$0000-$3FFF banco 0 (condizioni normali)
$4000-$7FFF* banco 1
$8000-$BFFF banco 2
$C000-$FFFF* banco 3

La routine Tre è molto importante, ma va usata con estrema cautela. Essa permette di modificare il banco, la posizione della memoria video e del generatore caratteri. Il primo parametro, compreso tra zero e tre, indica il banco da 16 Kb visto dal Vic II. Il valore di Par2, compreso tra zero e 15, corrisponde alle 16 posizioni possibili della memoria video all'interno dei sopracitati 16 Kb. Notate che se si sposta la memoria video bisogna aggiornare anche la locazione 648 in questo modo :

POKE 648,(PAR1*16384+PAR-2*1024)/256

perché altrimenti i comandi Print saranno indirizzati in una zona di memoria diversa da quella di dove si trova la memoria video (normalmente 648 contiene il valore quattro, infatti 4 x 256=1024, che è l'indirizzo della memoria video all'accensione del C64). Il terzo parametro, compreso tra zero e sette, indica la posizione del generatore caratteri o di un eventuale schermo in bit map; ogni unità corrisponde a salti di due Kb di Ram. Vediamo un esempio:

sys 49152,3:2,0,3:poke 648,128

Il Vic II vedrà i 16 Kb da 32768 a 49151, il video sarà posto a 32768 e sarà fissato il set minuscolo.

• Routine Quattro
sys 49152,4:PAR1

Questa routine riempie la finestra con il byte Par1, così si può utilizzare per cancellare il contenuto della finestra. Esempio:

sys 49152,4:85

riempie la finestra con un retino:

sys 49152,4:0

cancella il contenuto della finestra. Se la finestra non è stata attivata dalla routine Uno, non succede nulla, perché ogni routine prima di agire controlla il contenuto della locazione Flagon ($CD36). Questa locazione contiene sempre zero a meno è la routine Uno non sia già stata chiamata.

• Routine Cinque
sys 49152,5:PAR1

La routine disegna attorno alla finestra una cornice. Par1, compreso tra zero e tre, è il colore con cui viene disegnata. Zero è il colore dello sfondo, uno il colore definito con la routine Zero, mentre i colori due e tre sono disponibili solo in modo multicolore. Due e tre sono i colori contenuti in $D025 e $D026.

• Routine Sei
sys 49152,6:PAR1,PAR2,PAR-3,PAR4

Stampa un carattere sulla finestra. Par1, compreso tra zero e 511, è il numero del carattere prelevato dalla Rom caratteri del C64. Da uno a 26 ci sono le lettere maiuscole. Se Par2 vale zero, lo spazio sottostante il carattere viene cancellato e poi viene stampato il carattere. Invece se Par2 vale uno il carattere viene sovrapposto a ciò che già appare nella finestra. Par3 contiene l'ascissa del carattere sulla finestra. Ogni unità di ascissa corrisponde a otto pixel. Par4 è l'ordinata del carattere. Questa volta a ogni unità corrisponde un pixel. Un carattere è un quadrato di 8 x 8 pixel. Se il carattere risulta esterno alla finestra non viene stampato, oppure lo è solo in parte. Esempio:

sys 49152,6:3,0,0,2

Stampa una C nell'angolo in alto a sinistra. Il programma attinge le sue informazioni dalla Rom carattere ($D000), quindi se si vuole utilizzare un set personale di caratteri è necessario pokare il byte basso del nuovo indirizzo in ($C54E) e il byte alto in ($C554). Bisogna insomma comunicare il punto di inizio della nuova Ram caratteri.

• Routine Sette
sys 49152,7:PAR1,PAR2,PAR3

Questa è la routine di plot. Par1, compreso tra zero e 255, è l'ascissa relativa alla finestra, del punto che si vuole plottare. Par2 è invece l'ordinata dello stesso punto. Par3, compreso tra zero e quattro, è il colore. Zero corrisponde al colore di fondo, uno è il colore di base, mentre due e tre servono in modo multicolor.

Il valore quattro serve per testare la condizione fisica di un punto della finestra, cioè se Par3 =4 si può sapere di che colore è il punto [X;Y]. La locazione $CD3D (52541) contiene il valore del colore secondo le norme precedentemente descritte. Il punto [0;0] è l'angolo in alto a sinistra. Esempio:

sys 49152,7:5,10,1

Plotta un punto di coordinate [5;10] con il colore 1.

sys 49152,7:5,10,4:A=peek(52541)

Controlla il colore del punto [5;10], "A" contiene questo colore.

• Routine Otto
sys 49152,8:PAR1

Serve per eseguire l'Eor tra il valore Par1 e il contenuto della finestra. Permette di ottenere l'immagine inversa della finestra (Par1=255), ma può servire anche per scambiare tra loro i colori in modo multicolor (Par1=170). Esempio:

sys 49152,8:255

inverte la finestra.

• Routine Nove
sys 49152,9:PAR1,PAR2,PAR3,PAR4,PAR5

Ecco forse la routine più utile perché traccia una linea, dati gli estremi. Par1 è l'ascissa di inizio mentre Par3 è quella di arrivo. Par2 è l'ordinata di inizio e Par4 l'ordinata di arrivo. Par5 è il colore, già illustrato nella routine Cinque. Esempio:

sys 49152,9:0,20,0,40,1

Traccia una linea verticale lunga 20 pixel.

sys 49152,9:0,0,20,20,1

Traccia una retta inclinata di 45 gradi. Le righe vengono tracciate con continuità, ma se si pokano valori superiori a uno in $C8Fe (51454) la linea viene tratteggiata.

• Routine Dieci
sys 49152,10:PAR1

Dove Par1 è compreso tra zero e due. Se vale zero spegne la finestra, cioè disattiva sia gli sprite sia gli interrupt. Se vale uno disattiva solo gli sprite, dunque la finestra rimane ancora sullo schermo, solo che non si vede. In pratica vengono azzerati i bit della locazione $D015. Tutti i comandi di disegno modificheranno la finestra, ma noi non vedremo nulla. A questo punto per riattivarla basta usare la stessa routine con Par1 = 2. Le routine dalla Undici in poi funzionano solo se la finestra occupa una memoria contigua, cioè se non si sono modificati i valori della Tabbl, cosa che si può fare con delle poke.

• Routine Undici
sys 49152,11:PAR1,PAR2,PAR3,PAR4,PAR5

Questa routine permette il passaggio di dati dalla finestra alla grafica bit map e viceversa. Infatti la finestra ha un formato diverso dalla bit map. Par1 contiene il blocco da 1 Kb dove comincia la bit map. Esempio:

Tavola 1.

Configurazione di memoria di una finestra larga tre sprite e alta due

BASE1=peek(52619+0)
BASE2=peek(52619+1)

1	2	3
loc. BASE1	*loc. BASE1+1*64*	*loc. BASE1+2*64*
4	5	6
loc. BASE2	*loc. BASE2+1*64*	*loc. BASE2+2*64*

PAR1 =32

la bit map comincia a 32 x 1024, cioè a 32768. Se la bit map è già stata attivata il programma calcola automaticamente la sua posizione, basta porre Par1=0. Par2 contiene l'ascissa dello schermo in bit map (salti di otto pixel). Par3 è l'ordinata della bit map con salti di un pixel. Se Par4 contiene zero la bit map è copiata nella finestra, se invece contiene uno, il contenuto della finestra è trasferito nello schermo in bit map. Par5 contiene l'ordinata della finestra da cui comincia il trasferimento relativo alla finestra.

Questa routine, poiché trasferisce locazioni di memoria va usata con estrema cautela, perché se si sbagliano i conti si rischia di mandare in crash il sistema e dover resettare tutto.

• Routine Dodici

Con questa routine si possono trasferire aree di memoria come se fossero delle window. Cioè si può copiare una porzione di Ram nello spazio riservato alla finestra, o duplicare il contenuto della finestra altrove nella Ram.

sys 49152,12:PAR1,PAR2,PAR3,PAR4

Se Par1 vale uno, il contenuto della finestra, presente sullo schermo, viene ricopiato nella Ram a partire da:

PAR2*16384+PAR3*64

cioè nel banco Par2 e blocco Par3.

Se Par1 vale zero, la zona di memoria che comincia al banco Par2 e blocco Par3 viene ricopiata nella Ram occupata dalla finestra attuale. Il quarto parametro esprime il tipo di operazione logica da eseguire sulle due aree di memoria. Par4=0 ricopia i dati così come sono.

Se vale uno viene eseguito l'And, se vale due l'Or se invece vale tre avviene l'Eor. Esempio:

sys 49152,12:0,3,128,0

la finestra viene riempita con i dati che si trovano al banco=3 e blocco=128 3 x 16384+128 x 64=57344, e non avviene nessuna operazione logica.

sys 49152,12:0,3,128,2

esegue la sovrapposizione di due finestre.

• Routine Tredici

È in pratica la routine di Save, solo che si possono avere due possibilità :

1) sys 49152,13:4,"nome",device

Si usa se la finestra è attiva cioè presente sullo schermo e si vuole salvare proprio quella finestra. Device è il numero di periferica (uno registratore, otto disco).

2) sys 49152,13:PAR1,PAR2,PAR3, PAR4, "nome", device

Serve se si vuole salvare una zona di memoria come se fosse una window. Par1 è il banco da 16 Kb, Par2 è il blocco di memoria per quel banco. Par3 è la larghezza della finestra espressa in sprite, Par4 l'altezza in sprite. Par3 e Par4 corrispondono ai primi due parametri della routine Uno. Device indica ancora il dispositivo (uno-otto).

Durante il salvataggio la finestra viene automaticamente spenta, ma al termine del Save viene riattivata. Questo è necessario perché la presenza della finestra disturba le comunicazioni con le periferiche.

• Routine Quattordici

È praticamente la routine di Load, e si può usare in due modi diversi:

1) La finestra è attiva, e vogliamo caricare un'altra finestra nella stessa zona. Allora è sufficiente dare: sys 49152,14: 4,"nome",device.

L'altezza e la larghezza della finestra sono settati automaticamente, perché quando una finestra viene registrata, vengono salvati anche questi valori.

2) Si vuole caricare una finestra in una zona di memoria, lasciando attiva un'eventuale finestra presente sullo schermo. Allora:

sys 49152,14: banco, blocco, "nome", device

banco e blocco indicano la zona di memoria in cui ha inizio il caricamento.

Alla fine del caricamento, la finestra che precedentemente si trovava sullo schermo, viene riattivata. La larghezza della finestra si troverà nella locazione ($CD20), mentre la lunghezza in ($CD21).

Può succedere che nessuna finestra sia attiva e si provi a dare un sys 49152,14:4,"nome",device. Allora, poiché il programma non ha informazioni su dove mettere i dati della finestra che si desidera caricare, questa non sarà caricata, in pratica è come se non succedesse nulla.

Le routine Tredici e Quattordici possono essere usate tranquillamente in modo programma.

Sulla cassetta

Sulla cassetta allegata trovate quattro programmi dimostrativi oltre al programma Finestra già pubblicato sul numero precedente. Caricate per primo il programma Finestra con:

LOAD"FINESTRA",1,1 per il registratore e

LOAD"FINESTRA",8,1 per il drive.

Terminato il caricamento date un New e caricate i dimostrativi che sono: Demo1, Demo2, Demo3 e Demo4 e lanciateli con Run.

Nicola Chiminelli

SE PAPA' NON MI COMPRA AMIGA 500 CON I SUOI 4096 COLORI, DIVENTO NERA.
"Per anni ho avuto le mani impegnate con matite e colori, nella convinzione che artisti non si nasce ma si diventa. Poi, da un giorno all'altro e senza sforzo, il disegno non ha più avuto segreti per me. Non è una favola, è la realtà di Amiga 500. Con Amiga 500 carta e matite non mi servono più, perché posso disegnare direttamente sullo schermo, posso modificare i colori (ne ho ben 4096 a disposizione), posso ingrandire, ridurre, lavorare in prospettiva e in tridimensionale. Una soddisfazione così grande me la sono tolta a un prezzo che non mi ha mandato in rosso. Se poi un giorno volessi tradire il disegno per la musica, o la regia, o la narrativa, con Amiga 500 non avrei che l'imbarazzo della scelta. In più Amiga 500 è Commodore, cioè un capolavoro di prestazioni, di sicurezza e di affidabilità. Adesso che vi ho dipinto una realtà così rosa, correte subito a comprarvi Amiga 500 nel più vicino Commodore Point".
Per informazioni, dalle 14.00 alle 18.00
Hot-Line 02-66123237/40
Per sapere qual è il Point più vicino a casa vostra, telefonate allo 02/66123.1
Commodore
ONLY AMIGA MAKES IT POSSIBLE
PUBBLIMARKET/IDEA 2

I puntatori e le strutture di controllo sono due fra gli strumenti più potenti disponibili in Qpl. I primi permettono di risolvere brillantemente e senza fatica problemi di ogni genere e le seconde di utilizzare con successo la tecnica della programmazione strutturata e con il massimo dei benefici.

Gli strumenti del potere

In Qpl il puntatore è sicuramente un ottimo strumento, potente e tutt'altro che pericoloso e in questa puntata ne avrete la conferma. Non parleremo però solo di puntatori. Inizieremo finalmente a vedere più da vicino la struttura generale dei programmi Qpl e impareremo a usare tutti gli statement del linguaggio.

I puntatori

Prima di vedere la struttura generale di un programma Qpl proseguiremo il discorso interrotto la puntata precedente, che, se vi ricordate, riguardava i puntatori. I puntatori sono uno strumento potentissimo quanto sconosciuto a chi programma solo in Basic. In Qpl la gestione dei puntatori risulta abbastanza flessibile e semplice, quasi come in C e Pascal. Un puntatore è essenzialmente l'indirizzo di qualcosa, dove per qualcosa si intende l'oggetto cui il puntatore fa riferimento. L'utilità dei puntatori sta nel fatto che tramite essi si possono gestire molto facilmente strutture complesse prescindendo dalla organizzazione in memoria degli oggetti che formano la struttura stessa. In Qpl un puntatore è semplicemente una variabile di tipo word, cioè un intero assoluto di 2 byte, e quindi può essere trattato con gli operatori visti nella puntata precedente. Ci sono però alcuni operatori che sono pensati apposta per i puntatori. Eccoli:

• **#:** è l'operatore indirizzo. Può essere applicato soltanto a una variabile (non a un numero, a una stringa, a una costante o a una funzione) e ritorna l'indirizzo della variabile che segue. Questo operatore serve per indirizzare un puntatore su qualcosa. Vediamo un esempio:

```
word ptreal
word ptbyte
word ptint
word ptword
real q
byte w
int e
word r
...
ptreal=#q
ptbyte=#w
ptint=#e
ptword=#r
```

Le ultime quattro istruzioni assegnano a altrettanti puntatori (le variabili che compaiono alla sinistra del segno di uguale (=) sono infatti di tipo word) gli indirizzi di diversi oggetti. Precisamente assegnano ai quattro puntatori, in ordine, l'indirizzo di una variabile reale, di una variabile byte, di una variabile int e di una variabile word. L'utilità di avere un puntatore a un oggetto è data dalla presenza degli operatori descritti di seguito.

• **@<:** è un operatore di indirezione, cioè permette di accedere all'oggetto referenziato da un puntatore. Questo permette di accedere a un oggetto di tipo byte. Ecco un esempio:

```
word ptbyte
byte a
ptbyte=#a
if ptbyte@< > 5
...
```

In questo esempio ptbyte@< verrà trattato esattamente come se al suo posto ci fosse il contenuto della variabile a.

• **@-:** questo secondo operatore di indirezione permette di accedere a oggetti di tipo int (sempre tramite un puntatore).

• **@+:** è l'operatore di indirezione che permette di accedere a un oggetto di tipo word.

• **@.:** è l'ultimo degli operatori di indirezione e permette di accedere a un oggetto di tipo real.

Facciamo qualche altro esempio:

```
word ptr
data byte comdchar []='a','s','d','f','g'
...
ptr=#comdchar[3]
```

L'ultima istruzione assegna al puntatore ptr l'indirizzo del carattere comdchar[3] (è il carattere d).

```
byte buffer[80]
word ptr
byte char
...
ptr=buffer
...
char=ptr @<
```

L'ultima istruzione assegna alla variabile Char il primo carattere del vettore buffer.

Da notare che nell'assegnamento del puntatore non è stato necessario utilizzare l'operatore #. Questo succede perché il nome di un vettore viene considerato automaticamente un puntatore.

Tenete presente che non potete usare gli operatori di indirezione per modificare il contenuto della memoria (questo va a vantaggio dell'affidabilità dei programmi in cui si usano i puntatori, cioè non potete usare istruzioni del tipo:

```
ptr@< = 10
```

Per ottenere l'effetto desiderato dovete usare l'array predefinito m:

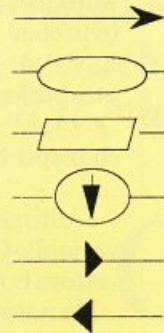
```
m[ptr]=10.
```

Variabili locali e non

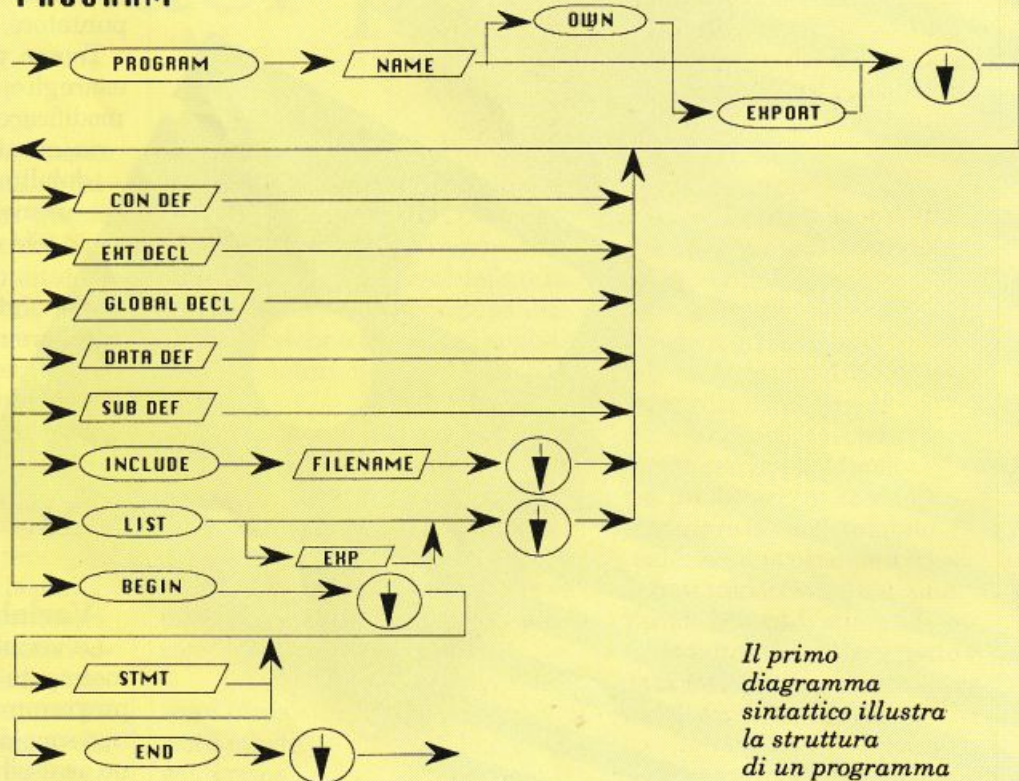
Le variabili sono dichiarate normalmente nella parte iniziale del programma. Le variabili di questo tipo sono definite globali perché sono accessibili in ogni parte del programma, anche all'interno delle singole subroutine. Al contrario le variabili dichiarate in una subroutine sono visibili esclusivamente in quella subroutine e non nel resto del programma. Ora che sapete formare delle espressioni e che avete imparato a dichiarare e a usare le variabili addentriamoci nella struttura vera e propria dei programmi Qpl.

La struttura dei programmi

I diagrammi sintattici sono una forma di descrizione della sintassi di un linguaggio che ha il grande pregio di risultare di immediata comprensione e estremamente concisa. Leggere i diagrammi sintattici è estremamente semplice. La parola che compare in neretto sopra ogni grafico indica la struttura che il grafico stesso definisce. Per esempio il diagramma sintattico etichettato con Programma (**tavola 1**) dice come si deve fare per scrivere un programma. Si deduce che un programma Qpl deve sempre iniziare con il simbolo terminale program seguito dal simbolo non terminale name, il nome del programma. Per ciò che deve venire dopo il nome del programma si hanno diverse alternative. Si può, per esempio, mettere il simbolo terminale own o export, oppure si possono omettere entrambi i simboli e inserire un Return.

Tavola 1.

A syntactically legal path.
A literal symbol, to be enterd as shown.
A user-supplied item based on another diagram.
End-of-line (Carriage Return or comment).
Indent one level.
'Exdent' (opposte of indent) one level.

PROGRAM

PROGRAM
NAME
OWN
EXPORT
CON DEF
EXT DECL
GLOBAL DECL
DATA DEF
SUB DEF
INCLUDE
FILENAME
LIST
EXP
BEGIN
STMT
END

Il primo diagramma sintattico illustra la struttura di un programma

Gli statement

Gli statement, o istruzioni, sono i componenti fondamentali di ogni programma dal momento che definiscono le azioni che il programma stesso deve effettuare. A differenza di quanto succede in Basic, in Qpl gli statement devono comparire in parti ben precise del programma. Per vedere in quale parte del programma potete mettere gli statement fate riferimento ai diagrammi sintattici. Ecco gli statement disponibili in Qpl:

• **Assegnamento:** è lo statement più semplice fra quelli disponibili ma è anche quello fondamentale. La sua struttura è, come in Basic:

variabile = espressione.

Variabile rappresenta il nome di una variabile mentre espressione può essere diverse cose: una costante, una variabile, una funzione, oppure una combinazione di questi oggetti in una espressione aritmetica o logica. Per la struttura precisa delle espressioni fate riferimento ai diagrammi sintattici. Lo statement di assegnamento assegna il valore dell'espressione alla variabile che compare alla sinistra del simbolo di uguale (=).

• **Strutture di controllo**: è una vera e propria classe di cui fanno parte:

- *if*: è la struttura di selezione conosciutissima anche a chi programma in Basic. In Qpl è disponibile una versione più potente che comprende anche la clausola else. La forma di questo statement è:

```
if espressione
        statement 1
        statement 2
        ...
else
        statement a
        statement b
        ...
```

Il funzionamento di questo statement dovrebbe essere piuttosto chiaro. Innanzitutto viene valutata l'espressione che compare dopo il simbolo terminale if e se il valore che si ottiene è true (un valore diverso da 0) vengono eseguiti tutti gli statement con indentazione maggiore (statement 1, statement 2 eccetera) che compaiono sulle linee successive. Se il valore dell'istruzione è false (0) allora vengono eseguite tutte le istruzioni che compaiono dopo il simbolo terminale else con indentazione maggiore di questo. Il ramo else è del tutto opzionale. Il punto di uscita di questa struttura di controllo è la prima istruzione non indentata rispetto a if e else (questi due simboli devono essere perfettamente incolonnati). Vediamo qualche esempio:

```
if x>10
        points=3
else
        points=1
score=score+point
...
```

Se x è maggiore di 10 saranno eseguite le istruzioni points=3, score=score+points e successive. Se x è minore o uguale a 10 allora saranno eseguite le istruzioni points=1, score=score+points e successive.

Questo esempio mostra chiaramente come sia possibile innestare più if per testare condizioni mutuamente esclusive.

```
if char='d'
        draw
else if char='e'
        erase
else if char='q'
        exit
else
        output "Illegal command."
```

Ecco un esempio un po' più complesso che servirà anche a dimostrare l'efficacia dell'uso dell'indentazione:

```
if x>100                    :1
        if y>x              :2
                z=3+x       :3
                y=0         :4
        else                :5
                y=1         :6
                if x>200    :7
                      z=y-100 :8
q=y+z                       :9
```

Se l'espressione valutata nello statement 1 (x>100) è falsa verrà eseguito lo statement 9 (q=y+z). Se invece risulta vera verrà eseguito lo statement 2.

Questo a sua volta controlla gli statement 3 e 4, che saranno eseguiti solo nel caso in cui y>x, e lo statement 5, che sarà eseguito se y<=x. Le istruzioni da 6 a 8 fanno parte dell'else alla linea 5.

-*while*: è una struttura di iterazione. La sua forma è:

```
while espressione
        statement 1
        ...
```

e si comporta in questo modo: se il valore dell'espressione che compare dopo while è true allora saranno eseguiti tutti gli statement successivi con indentazione maggiore, quindi si ritornerà a valutare l'espressione. In caso contrario l'esecuzione riprenderà con il primo statement con la stessa indentazione di while.

- *repeat*: è abbastanza simile all'istruzione precedente. Eccone la forma:

```
repeat
        statement 1
        ...
until espressione
```

Ecco come si comporta: vengono eseguiti tutti gli statement successivi a repeat con indentazione maggiore. Quindi viene valutata l'espressione che compare dopo until e se il risultato è false allora l'esecuzione riprende da repeat mentre se è true continua dal primo statement dopo until.

- *for*: è analogo al for del Basic e ha la seguente struttura:

```
for indice = min to max
        statement 1
        ...
```

indice è una variabile (non può essere l'elemento di un vettore) ed è sempre di tipo word. Min e max sono due espressioni e controllano il numero di iterazioni da eseguire. Le istruzioni interne al ciclo sono sempre eseguite almeno una volta, anche se min>max.

- *choose*: è simile al case del Pascal. Questo statement ha la seguente struttura:

```
choose espressione
val 1
```

```
        statement a
        ...
val 2
        statement j
        ...
val 3
        statement x
        ...
...
else
        statemnt k
        ...
```

Il funzionamento è abbastanza semplice. Viene valutata l'espressione che compare dopo choose e quindi il valore ottenuto viene confrontato con val 1, val 2 eccetera.

Se si trova un valore uguale a quello dell'espressione vengono eseguite tutte le istruzioni successive a quel valore con indentazione maggiore. In caso contrario vengono eseguiti gli statement dopo else. Ecco un esempio:

```
choose getc
'b'
        x=0
        start
'c'
        contin
'l'
        x=9999
        lastline
else
        put "Illegal letter"
...
```

L'espressione che compare dopo choose non può essere di tipo real e tutti i valori che figurano in corrispondenza delle alternative devono essere dello stesso tipo (per evitare errori usate gli operatori di conversione). Ecco un altro esempio:

```
word num
...
choose num
1:+
        statement 1
        ...
2:+
        statement a
        ...
...
```

se non mettete gli operatori di conversione il compilatore segnalerà un errore.

- *break:* permette di uscire da un ciclo while o repeat (ma non for). Ecco un esempio:

```
while true
        x=x+1
        if x>1
                break
```

- *next*: ha una funzione facilmente intuibile. Va usata all'interno di un ciclo while o repeat (ma non for) e causa un salto all'inizio del ciclo.

- *nothing*: questo statement non fa assolutamente nulla. Si tratta infatti dello statement vuoto. Va usato nei cicli in cui non si vogliono inserire istruzioni. Ecco un esempio:

```
repeat
        nothing
until getc=cr
```

questo frammento di programma testa la pressione di un tasto. Se non usate nothing all'interno di un ciclo vuoto il compilatore vi segnalerà un errore.

• **Escape e refuge:** questi due statement sono assolutamente originali e non hanno eguali in altri linguaggi. Permettono di saltare da un punto del programma a un altro punto qualsiasi, indipendentemente dal livello in cui si trovano la posizione di partenza e di arrivo del salto (per esempio, dall'interno di una subroutine potete saltare in un' altra subroutine o al programma principale). Questa coppia di istruzioni si rivela particolarmente utile per trattare situazioni critiche.

Refuge serve per identificare il punto di arrivo del salto e escape per effettuarlo. Ecco la sintassi delle due istruzioni:

```
refuge n
```

in questo modo lo statement successivo a escape viene etichettato con la label n

```
escape n
```

salta all'istruzione successiva a refuge n.

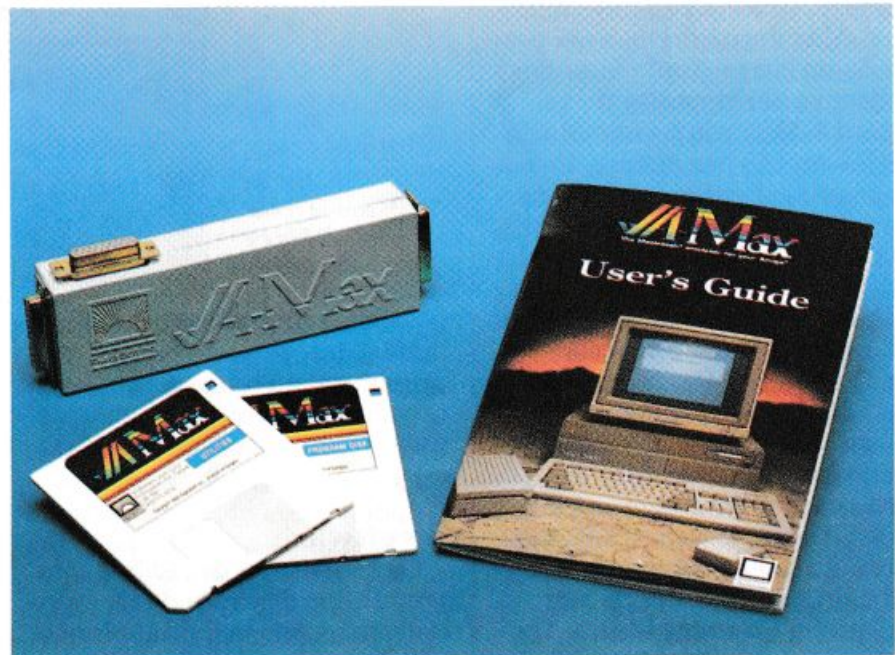
Ecco un esempio di applicazione:

```
proc error ; visualizza il messaggio
d'errore e esce
arg word errno
...
begin
put nl,errormsg[errno],nl ; visualizza il messaggio d'errore
escape 1
end

proc checkchar
begin
if char <> legal
        error 3
...
end

proc doline
...
refuge 1 ; salta in questo punto dalla proc error
while getl(line)
...
```

Daniele Maggio
(continua)

Il software a cui l'articolo si riferisce è comparso sui numeri 5 (giugno), 7 (settembre) e 8 (ottobre) di Radio Elettronica & Computer. *Il software funziona solo su disco ed è costituito da due programmi (Qpl1 e Qpl2) che, lanciati, trasformano un disco da formattare nel disco di sistema di Qpl.*

Falso d'autore

Per quanto riguarda l'emulazione di altri sistemi, Amiga non ci ha certamente abituato bene. La velocità di elaborazione del sitema emulato, infatti, risultava sempre insufficiente e l'emulazione inutile. Nel caso delle schede di emulazione Xt e At la cifra proibitiva rendeva l'operazione paragonabile all'acquisto di un nuovo sistema. Così quando qualche mese fa ci giunse la notizia che la software house canadese, ReadySoft, aveva prodotto A-Max, un emulatore Macintosh per Amiga, prendemmo la notizia con le pinze: la speranza di un prodotto valido c'era, ma non era ancora il momento di gioire. Funzionerà o non funzionerà? Il dubbio ci ha tenuto in sospeso per il tempo necessario a procurarci l'accessorio, a leggere al volo il manuale di istruzioni, installare l'apparecchio e capirne il funzionamento. Ebbene il risultato della prova ha lasciato a bocca aperta l'intera redazione del Gruppo Editoriale Jce: poco mancava che qualcuno ci chiedesse perché Amiga era stata sostituita con un Macintosh, data l'incredibile fedeltà di funzionamento. Ma procediamo con ordine.

L'evoluzione di Amiga sembra veramente inarrestabile. Da oggi, infatti, tutti gli utenti di questo personal possono avvalersi anche di tutto il software scritto per Macintosh grazie ad A-Max, un efficacissimo emulatore prodotto da una casa canadese

L'hardware

Come potete vedere dalla fotografia di apertura l'hardware consiste in una piccola scatola grigia con tre connettori. Il connettore maschio deve essere inserito nella porta drive, mentre i due connettori femmina servono per eventuali drive aggiuntivi di tipo Amiga o Apple. All'interno della scatola si trova un circuito stampato a doppia faccia realizzato con molta cura, pochissima componentistica e due zoccolini per le Rom Apple. Per ovvi motivi di Copyright, non è stato possibile copiare il sistema operativo di Macintosh, pertanto, per poter utilizzare l'emulatore, è necessario acquistare le Rom originali. Esistono due possibilità: le Rom da 64 Kb del vecchio Macintosh oppure le Rom da 128 Kb del Macintosh Plus. È decisamente consigliabile optare per la seconda possibilità, poiché le Rom da 64 Kb non permettono l'utilizzo del software commerciale più diffuso per Macintosh. L'acquisto delle Rom, comunque, non è un problema poiché è possibile trovare A-Max già fornito dei due preziosi chip, naturalmente, originali Apple. L'installazione di A-Max è molto semplice: si apre la scatoletta svitando due viti, si inseriscono le Rom con molta delicatez-

za, si richiude, si inserisce A-Max nel retro di Amiga, nella porta drive. Se si dispone di drive aggiuntivi è necessario collegarli sul retro di A-Max, a meno ché, tali drive, abbiano il bus passante nel qual caso è possibile inserire A-Max dopo i drive. Se si possiede un drive Apple lo si deve collegare nell'apposita porta sul lato di A-Max, e lo pseudo-Macintosh è pronto per partire. Prima di passare al software è utile parlare dei problemi di compatibilità tra i drive Commodore e Apple. I drive Amiga non possono leggere direttamente i dischetti formattati dal Macintosh a causa di una profonda differenza hardware. I drive Apple formattano i dischetti utilizzando differenti velocità di rotazione: i drive Amiga, così come i drive Ibm e i drive Atari, hanno la velocità di rotazione fissa, e non è possibile modificare tale velocità via software, quindi l'incompatibilità tra i drive è totale. Poiché per utilizzare Macintosh è necessario disporre dei programmi, i tecnici della ReadySoft hanno pensato a diverse soluzioni. Innanzitutto A-Max è in grado di inizializzare i dischetti da tre pollici e mezzo utilizzando i drive Amiga secondo un formato simile al formato Macintosh che chiameremo formato A-Max. Questo significa che è possibile formattare a doppia o a singola faccia, ottenendo dischetti da 800 Kb oppure 400 Kb equivalenti ai dischetti Macintosh. Resta però il problema di trasferire il software Macintosh sui dischetti in formato A-Max. La soluzione migliore consiste nell'utilizzare un drive Apple per copiare i programmi dal formato Macintosh al formato A-Max. È sufficiente collegare il drive Apple a A-Max e copiare i dischetti come si procede normalmente con Macintosh. Se non si dispone di un drive Apple, ma si ha accesso a un Macintosh, è ugualmente possibile effettuare il trasferimento, ma si deve seguire una procedura più lunga e complessa che descriveremo parlando del software in dotazione a A-Max.

Fase 1. Il Workbench di Amiga con le icone di A-Max e alcune utility

Il software

La confezione di A-Max comprende due dischetti da tre pollici e mezzo, uno in formato Amiga e l'altro in formato ibrido A-Max Macintosh. Il dischetto Amiga comprende il software di start-up, ovvero è un disco Workbench 3.0. Oltre ai vari tool di Workbench sono presenti due programmi dedicati alla gestione di A-Max: A-Max Startup e File Transfer. Amiga non si accorge della presenza di A-Max finché non si seleziona da Workbench o da Cli il programma di start-up. È quindi possibile lasciare A-Max sempre collegato ad Amiga, senza che nasca alcun problema. Selezionando A-Max compare il menù di selezione. Da questo menù è possibile scegliere alcuni parametri di emulazione. Il secondo disco A-Max è formattato in modo decisamente inconsueto: una facciata è in formato A-Max, l'altra è in formato Macintosh. Inserendo questo disco in Amiga con A-Max leggiamo 400 Kb di dati in formato A-Max. Inserendolo in Macintosh ne leggiamo altri 400 Kb in formato Mac. Su questo dischetto si trovano alcune utility per il trasferimento di file da Macintosh a A-Max, nonché da A-Max ad Amiga. In particolare chi non dispone di un drive Apple per effettuare il trasferimento dei programmi al formato Amiga può servirsi di un Macintosh per formattare i dischetti di trasferimento. Per dischetto di trasferimento si intende un floppy disk formattato a soli 272 Kb con la particolare proprietà di essere scrivibile da Macintosh e leggibile da A-Max. Tramite i dischetti di trasferimento è possibile copiare sia singoli file Macintosh, sia interi dischetti. La procedura è piuttosto lunga e noiosa, tuttavia funziona perfettamente.

Il video

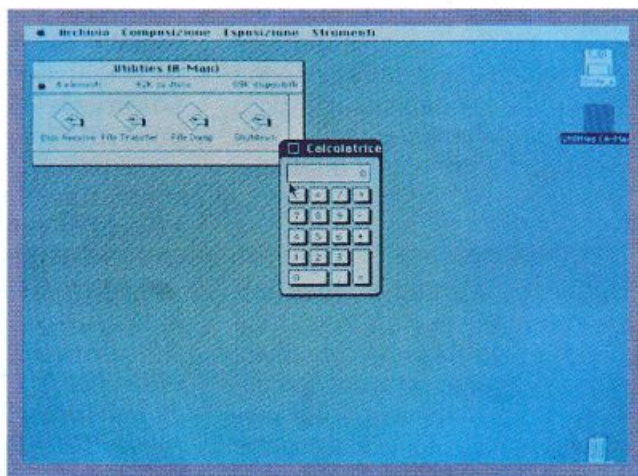
Il video standard di Macintosh ha una risoluzione di 512 x 342 pixel, mentre Amiga nella versione Pal ha una risoluzione orizzontale di 320 o 640 pixel e una risoluzione verticale di 256 pixel oppure 512 pixel in interlace. Con i nuovi Chip Ecs (Extended Chip Set) che la Commodore sta per mettere in commercio è possibile ottenere da Amiga una risoluzione verticale di 480 pixel in modo non interlacciato e ben 960 pixel in modo interlacciato. Se invece si possiede la scheda A2024 è possibile raggiungere la fanta-

stica risoluzione di 1008 x 800 pixel. Sono dunque possibili numerosi modi di emulazione. Chi possiede i nuovi chip o la scheda A2-024 può sfruttare al massimo il proprio hardware e si ritrova con un Macintosh con monitor professionale. Se viceversa si dispone di un'Amiga standard è possibile scegliere tra il modo interlacciato o non interlacciato. Se si vuole emulare lo schermo di Macintosh in modo fedele è necessario scegliere il modo Mac che fornisce su un video interlacciato la stessa risoluzione di Macintosh. È conveniente selezionare la risoluzione massima disponibile, ossia 640 x 512 pixel. Naturalmente il video interlacciato presenta i noti problemi di flickering, il fastidioso tremolio di schermo. Si può ridurre in modo drastico il fenomeno tramite un'accurata scelta dei colori di schermo. A-Max emula un Macintosh con monitor monocromatico. Tramite un'altra opzione del menù di avvio, è possibile selezionare tra i colori di default di A-Max e i colori di sistema. Chi non sopporta il modo interlacciato può utilizzare la risoluzione di 640 x 256 pixel e in tal caso per vedere l'intero schermo di Macintosh ci sono due possibilità: uno tra lo schermo che scrolla seguendo il mouse, oppure si può dividere lo schermo in due parti selezionabili tramite il tasto destro del mouse. In entrambi i casi ci pare che la simulazione sia poco efficace e soprattutto poco comoda: lo schermo interlacciato offre di sicuro risultati migliori.

La memoria

A-Max è in grado di sfruttare l'intera memoria Ram di Amiga durante l'emulazione Macintosh, ma si deve prestare attenzione ad alcuni particolari. La memoria Ram di Macintosh inizia alla locazione zero e costituisce un blocco unico continuo. Amiga invece ha una struttura più complessa, infatti la Chip Ram consiste di 512 Kb disposti a cominciare dalla locazione zero, mentre le espansioni di memoria sono allocate a indirizzi diversi non contigui alla Chip Ram. Amiga 2000 e Amiga 500 con l'espansione di memoria da 512 Kb hanno 512 Kb di Fast Ram disposta dopo la locazione $C00000. Se l'Amiga dispone solamente della Chip Ram non ci sono problemi di emulazione, tuttavia si deve tenere presente che A-Max occupa 128 Kb di memoria, pertanto si dispone di un Macintosh con soli 384 Kb di Ram. Decisamente troppo pochi per potere utilizzare i potenti pacchetti software disponibili sul mercato. Se invece si possiede un'Amiga espansa a 1 Mb in teoria si ottiene Macintosh con 896 Kb di Ram. In questo caso, con alcuni programmi, possono sorgere dei problemi di compatibilità a causa della suddivisione della memoria in due blocchi distinti. In realtà tali problemi sono molto limitati, e comunque è possibile aggirarli in diversi modi. Il menù di avviamento di A-Max dispone di alcuni comandi che permettono di configurare la memoria durante l'emulazione di Macintosh. Se si sceglie l'opzione "No Expansion" qualsiasi espansione di memoria viene ignorata, quindi si ottiene il massimo grado di compatibilità con Macintosh a spese di una quantità di Ram molto ridotta. Con l'opzione "No $C00000" si può eliminare l'espansione di memoria di Amiga 500, nonché i 512 Kb di Fast Ram di Amiga 2000. In questo modo però A-Max si installa ugualmente nella Fast Ram. Così facendo otteniamo un Macintosh con 512 Kb di memoria e nessun problema di compatibilità. Scegliendo invece l'opzione User Memory si utilizza tutta la Ram disponibile durante l'emulazione. La memoria non utilizzata per l'emulazione comunque non va sprecata poiché A-Max la trasforma automaticamente in Ram Disk. È possibile rimediare al problema della differente configurazione di memoria anche in modo completo e definitivo, agendo via hardware. Se si possiede un'Amiga 2000 è possibile cambiare la disposizione della Ram spostando due ponticelli sulla piastra madre. Si tratta di un'operazione semplicissima che dispone l'intero megabyte di Ram di Amiga 2000 non espansa in un unico blocco a partire dalla locazione zero, come in Macintosh. In questo modo si ottiene un Macintosh con 896 Kb di Ram senza alcun problema di compatibilità dovuto alla memoria. È possibile effettuare la medesima

Fase 2. Il pannello di controllo del programma. Il passaggio successivo trasforma completamente il sistema

Fase 3. La scrivania di Machintosh con il contenuto del disco utility di A-Max e la solita calcolatrice

operazione anche su Amiga 500, anche se l'intervento è più complesso della rimozione di un ponticello, poiché è necessario tagliare una pista del circuito stampato. Questa modifica hardware è particolarmente utile durante l'emulazione di Macintosh, tuttavia crea alcuni problemi ad Amiga in Amiga Dos: il sistema operativo vede 1 Mb di Chip Ram, mentre Agnus, il chip custom che gestisce la memoria video di Amiga, può accedere solamente ai primi 512 Kb. È possibile rimediare a questo ulteriore problema inserendo nei propri dischi di sistema il comando KillChip, il cui file si trova sul disco di A-Max. Questo comando, che deve essere inserito nella startup-sequence, informa Amiga che solo i primi 512 Kb di Ram costituiscono la Chip Ram, risolvendo ogni problema. Quando saranno disponibili i nuovi chip custom, ossia l'Ecs, questo problema sarà del tutto risolto poiché la Chip Ram ammonterà a 1 Mb e quindi la difficoltà sarà risolta a monte. Insistiamo, comunque, nell'affermare che questi problemi nascono solamente se si desidera sfruttare al massimo A-Max: noi abbiamo potuto usare senza difficoltà tutta la Ram di Amiga senza apportare modifiche e senza incontrare alcun problema.

Amiga 1000

L'ultima raffinatezza di A-Max in tema di memoria è dedicata ai proprietari di Amiga 1000. Infatti A-Max non ha alcun bisogno del KickStart durante l'emulazione. I 256 Kb di Ram utilizzati da Amiga 1000 per il KickStart possono essere aggiunti alla memoria di sistema selezionando l'opzione "Use KickStart".

Stampante

L'ultima opzione del menù di avvio è riservata alla gestione della stampante. Le stampanti Apple utilizzano uno standard diverso dallo standard Epson o Ibm utilizzato dalle stampanti solitamente impiegate in abbinamento ad Amiga. A-Max è in grado di emulare anche le stampanti Apple, convertendo i codici Image Writer in codici Epson. Sono possibili differenti modi di emulazione. Se si possiede una stampante a nove aghi è necessario selezionare l'emulazione di Image Writer 9 Pin e si deve disporre del driver Macintosh per la Image Writer. Se, invece, si possiede una 24 aghi è possibile utilizzare l'opzione Image Writer 24 Pin con il driver Image Writer, nel qual caso si ottiene una stampa Lq effettuata con un doppio passaggio a otto aghi, oppure si può selezionare l'emulazione Lq 24 con il driver Lq Image Writer. Quest'ultima possibilità offre i risultati migliori, ovvero una stampa Lq in singola passata che sfrutta i 24 aghi. La qualità di stampa è comunque molto alta, tuttavia sorgono alcuni problemi. La differenza più fastidiosa consiste nel differente formato delle testine Image Writer rispetto allo standard Epson. Con il driver Image Writer si ottengono stampe più strette del dovuto, a causa della maggiore densità orizzontale delle stampanti Epson. Se invece si utilizza il driver Lq Image Writer si ottengono pagine più lunghe di circa il 20% a causa della minore densità verticale dello standard Epson 24 aghi. Altri problemi sono dati dalla gestione delle pagine, tuttavia è sufficiente un poco di pazienza per ottenere un'emulazione molto efficace e i problemi da noi rilevati sono ben poca cosa rispetto ai vantaggi offerti dall'emulatore.

Il manuale

Con un prodotto così sofisticato è importante valutare attentamente anche il manuale: in questi casi la mancanza di istruzioni può penalizzare decisamente il valore di un prodotto. Il manuale di A-Max è sufficientemente completo e spiega accuratamente tutte le possibilità offerte dall'hardware e dal software.

Manca, invece, qualsiasi cenno al funzionamento di Macintosh, pertanto oltre al disco di sistema e ai programmi è utile procurarsi un manuale sul sistema operativo Macintosh.

L'emulazione

Dopo tutte queste parole è davvero ora di vedere come lavora Amiga durante l'emulazione Macintosh. Vi abbiamo già prean-

nunciato il buon funzionamento di A-Max: vediamo ora qualche particolare. Per utilizzare A-Max è necessario disporre di un disco di sistema Macintosh. Poiché la Apple ha l'ottima abitudine di rivedere continuamente il proprio sistema operativo al fine di ottenere un risultato sempre più valido, esistono numerose versioni del disco di sistema.

La ReadySoft afferma che tutte le versioni fino alla 6.0.2 possono funzionare correttamente con A-Max. Noi abbiamo provato dapprima la 5.0, poi, visto il successo della prova, siamo passati direttamente alla nuova 6.0.3. Abbiamo utilizzato intensamente A-Max con entrambe le versioni di sistema per una settimana e non siamo riusciti a trovare davvero nessun problema. Passiamo ai pacchetti software.

La Readysoft afferma che con A-Max si possono utilizzare, tra gli altri, i seguenti programmi: HyperCard, PageMaker 1.2, Word, Excel, MacTerminal, MacWrite, MacDraw, MacPaint. Ebbene abbiamo provato tutti questi programmi con un'Amiga 2000 e con un'Amiga 500 espansa a 1 Mb, e nessuno di questi ha dato alcun problema, neppure utilizzando interamente la Ram. Solo Excel necessita l'opzione "No $C0000", altrimenti rifiuta di avviarsi. Anche Multifinder, il sistema multitasking di Macintosh, funziona correttamente, tuttavia per poter essere utilizzato efficacemente richiede una maggiore quantità di Ram. Abbiamo provato questi programmi sia su Macintosh Plus, sia su Amiga, in modo da apprezzarne le differenze, e vi possiamo davvero dire che non vi sono state differenze apprezzabili. L'unico particolare degno di nota consiste nella differente risoluzione: da una parte Amiga presenta il noto difetto del flickering di schermo, dall'altra, però, offre uno schermo più grande decisamente più comodo. Per quanto riguarda il funzionamento è particolarmente interessante il risultato di un test effettuato con QuickBasic. Abbiamo fatto girare il programma seguente su Amiga con A-Max e su Macintosh Se, senza compilarlo:

```
a=2.6342
FOR i=1 TO 10000
b=SIN(a)
NEXT i
```

Risultato sorprendente: Amiga esce dal ciclo dopo 34 secondi. Circa quattro secondi dopo arriva annaspando Macintosh.

Abbiamo anche provato a testare la velocità di tracciatura grafici con il seguente programma, sempre tramite QuickBasic:

```
FOR i=1 to 100
CIRCLE (200,200),i*2
NEXT i
```

Anche questa prova ha fornito un risultato eccellente poiché in sette secondi sono stati tracciati i cento cerchi di raggio variabile.

• Word 4.0

Il risultato più eclatante comunque è stato fornito da Word 4.0, la nuova versione per Macintosh del potentissimo word processor della MicroSoft. Ebbene non solo Word funziona in modo eccellente, ma supera in larga misura le prestazioni fornite da qualsiasi word processor per Amiga Dos. Non è questa la sede di raccontare la potenza di questo word processor, sicuramente uno dei migliori, se non il migliore, sul mercato dei personal computer. Quello che ci ha davvero meravigliato è il fatto che Word funzionando su Amiga in emulazione Macintosh, fornisce risultati decisamente migliori di qualsiasi word processor scritto appositamente per Amiga. Questo stesso articolo è stato redatto con Word 4.0, Amiga e A-Max.

• QuickBasic

Anche QuickBasic fornisce risultati migliori di Amiga Basic e di Ac Basic Compiler, soprattutto in termini di gestione delle finestre di lavoro, di scrolling dei listati e di comodità d'uso. Naturalmente si deve considerare che si ha a che fare con uno schermo monocromatico.

Tirando le somme

Dire che il risultato della prova di A-Max sia stato sorprendente significa ridurre il valore di questo oggetto che, a un prezzo relativamente modesto, offre praticamente tutta la potenza di Macintosh Plus. A-Max è anche la prova del notevole valore dell'hardware di Amiga, della sua potenza e flessibilità.

A nostro avviso è anche la prova della scarsa efficenza del sistema operativo di Amiga, infatti Amiga con A-Max ci sembra un computer più potente ed efficente di Amiga con Amiga Dos, almeno per quanto riguarda l'uso professionale del computer, non solo per la maggiore professionalità e potenza dei pacchetti software di Macintosh, ma anche semplicemente a livello di sistema, di velocità di gestione delle applicazioni e dei file.

Se possedete un'Amiga 500, 1000 o 2000 con almeno 1 Mb di memoria e mezzo milione che vi avanza non potete privarvi di questo preziosissimo accessorio.

Gianni Arioli

A-Max ci è stato gentilmente fornito dalla Newel Srl, via MacMahon 75, Milano. Tel. 323492. È disponibile presso i migliori rivenditori.

Cosa, Come, Quanto?

Notizie, novità e prezzi da tutto il mondo

Prodotti Logitek

Iniziamo questa nuova rubrica presentadovi alcuni prodotti distribuiti in Italia dalla Logitek. La Logitek è un Commodore Point (autorizzato dalla Commodore Italiana) utilizzato in particolare dalla Commodore per dimostrazioni nel campo Desktop Publishing e Desktop Video.

Il primo accessorio è uno **scanner da tavolo** in formato A4, che può funzionare anche come fotocopiatrice.

Si tratta di un accessorio in grado di trasformare qualsiasi documento, sia esso un foglio singolo o una pagina di libro, in un file Iff. Ricordiamo che il formato Iff (Interchange file format) è lo standard utilizzato da Amiga per registrare su disco qualsiasi immagine. Tale standard è adottato naturalmente da tutti i pacchetti software grafici, quali per esempio le varie versioni di De Luxe Paint, nonché dal software Dtp. Proprio in questo tipo di applicazione lo scanner da tavolo trova il suo ambiente di lavoro ideale. Vediamo qualche particolare tecnico: la lettura dei documenti è effettuata tramite Ccd (Charge coupled device), lo stesso procedimento utilizzato per le telecamere portatili. La risoluzione massima ammonta a 200 dpi (punti per pollice) pari a circa otto punti per millimetro. Il costo di questo accessorio è all'altezza delle prestazioni fornite: un milione 450mila lire.

Ancora per il Desktop Publishing è disponibile **Handy Scanner**. Si tratta di un prodotto più semplice e più economico: la risoluzione è sempre di 200 dpi, tuttavia la larghezza di scansione è di soli 64 mm. Handy Scanner è disponibile in due versioni: il tipo 2 in bianco e nero, che costa 495mila lire e il tipo 3 che digitalizza con 16 toni di grigio e costa 780mila lire. È possibile utilizzare Handy Scanner con due tipi di software: Handy Painter è un editor grafico tipo De Luxe Paint, mentre Handy Reader è un riconoscitore di testi. In particolare Handy Reader è un prodotto molto evoluto, infatti utilizzato in abbinamento a Handy Scanner o a un altro scanner è in grado di leggere testi scritti anche con differenti font carattere, stampati oppure scritti a macchina, e di convertirli in codici Ascii leggibili da qualsiasi word processor. Nel campo del Desktop Video la Logitek offre una apparecchiatura dalle caratteristiche marcatamente professionali: il **Genlock Vcg 3**. Questa apparecchiatura è il ponte di collegamento tra il mondo digitale di Amiga e il mondo analogico delle apparecchiature video: telecamere, videoregistratori e monitor. Con Vcg-3 è possibile registrare il segnale di Amiga generato tramite programmi di animazione quali Provideo Plus, Tv Text 3D, i vari programmi della serie De Luxe e il fantastico Sculpt 4D, inoltre è possibile sovrapporre il segnale di Amiga a un segnale video per effettuare titolazioni e così via. La Logitek offre anche un pacchetto completo di hardware e software adattissimo a uno studio video professionale.

Ditta Norma

Dall'Austria, paese così poco noto al mondo dei computer, giunge un utilissimo accessorio per C64: un multimetro elettronico. Si tratta di una cartuccia dall'aspetto simile ai vari velocizzatori, sprotettori e fa tutto quello cui ormai siamo abituati, tuttavia il Normameter Mp 11 è uno strumento completamente differente. Lo Mp 11, prodotto dalla Norma di Wiener Neudorf, trasforma il C64 (e naturalmente il C128) in un completo multimetro in grado di misurare voltaggi, correnti, temperature e frequenze. Pensiamo che sfruttando adeguatamente le capacità del C64 insieme a questo accessorio si possa ottenere uno strumento molto interessante e utile.

Commodore

Una giuria composta dai lettori di due prestigiose riviste americane nel settore video communication, *AV Video* e *Video Manager*, ha proclamato **Amiga 2000** "prodotto dell'anno". La scelta di tale computer è stata motivata in modo molto sintetico, ma significativo: "Amiga 2000 è il personal più utile nel lavoro". Amiga 2000 è stata premiata a Las Vegas durante l'annuale Salone della National Association of Broadcasters. Sempre la Commodore ci informa della nascita di un nuovo computer Ms-Dos, il **Pc 30-III**. Si tratta di una macchina

basata sul microprocessore 80286 funzionante con un clock variabile da 6 a 12 MHz, che naturalmente può essere coadiuvato dal fratello esperto in matematica 80827. La memoria Ram ammonta a soli 640 Kb, ma può essere espansa fino a 15 Mb. Dispone di un floppy disk drive da tre pollici e mezzo ad alta densità che può formattare dischetti fino a 1440 Kb e di un hard disk da 20 Mb. Il video è controllato da una scheda grafica Ega Wonder 800 che può emulare i modi Cga, Mda e Hercules. È nata una collaborazione tra la Commodore stessa e la Scuola Radio Elettra finalizzata allo sviluppo della didattica nel settore informatico. Lo scopo di questa cooperazione è l'insegnamento di Ms-Dos e dei tre pacchetti software più diffusi: Wordstar, dBase III Plus e Lotus 1-2-3. La Commodore contribuisce con la propria esperienza in campo informatico e con il suo Pc 10 III, un compatibile Ibm basato sul microprocessore 8088 dal costo piuttosto contenuto, particolarmente adatto a chi desidera addentrarsi nel mondo dell'informatica Ms-Dos. La **Scuola Radio Elettra** naturalmente offre la trentennale esperienza in campo didattico. Il metodo di insegnamento utilizzato è di tipo Cai (Computer adder instruction), ovvero è il computer stesso che si incarica di fornire allo studente l'aiuto e le nozioni necessarie per procedere nell'apprendimento della materia. Questo metodo si basa su una serie di programmi interattivi, ovvero programmi in cui lo studente non subisce passivamente le istruzioni del computer, ma è spinto a collaborare e quindi a sfruttare la macchina. Gli studenti comunque non sono abbandonati a sè stessi poiché la Scuola Radio Elettra ha aperto una rete di Open Center dove è possibile esercitarsi con l'aiuto di un insegnante. Il programma della Commodore e della Scuola Radio Elettra appare particolarmente interessante poiché mira a fare conoscere i programmi commerciali più diffusi, e soprattutto pone lo studente in condizione di sfruttarli efficacemente.

La **Commodore International** ha aumentato in modo significativo il proprio utile. In particolare ha subito un notevole aumento la domanda di Amiga e di Ms-Dos. Ci sembrano molto significativi questi dati: il 40% delle vendite Commodore è costituito dai modelli Amiga, il 20% dai compatibili Ibm e il restante 40% è ancora in mano ai C64 e C128. Anche se questi ultimi sono di gran lunga i meno potenti tra i diversi prodotti Commodore, essi sono considerati il massimo successo, il prodotto storico della Commodore, poiché hanno raggiunto la ragguardevole cifra di nove milioni di macchine installate, di cui ben un milione è stato venduto lo scorso anno. Niente male per degli otto bit con il clock di 1 MHz in questi anni in cui si parla sempre più di 32 bit a 10, 20 e più MHz!

Per trasferire i programmi di RE&C

Molti lettori hanno incontrato difficoltà nell'eseguire le operazioni di trasferimento dei programmi da nastro a disco. L'utility Dsave richiedeva infatti di specificare due indirizzi che definivano la zona occupata dal programma da trasferire. La sequenza di operazioni non era sempre uguale per tutti i programmi, perciò molti lettori inesperti non hanno potuto operare tutti i trasferimenti desiderati. In questo numero è disponibile sulla cassetta allegata una nuova utility denominata Dsave v2.

Dopo il caricamento del solito menù all'inizio della cassetta, tutti coloro che vorranno trasferire i programmi sul disco dovranno caricare e lanciare Dsave v2. Il menù offre tre possibilità:

1 *- La cassetta verrà letta e il primo programma incontrato caricato. A questo punto viene chiesta conferma per il trasferimento sul disco, dopodiché si passerà al caricamento del successivo programma sulla cassetta e così via.*

2 *- Scegliendo la seconda opzione, invece, verranno salvati su disco tutti i programmi automaticamente, senza selezioni da parte dell'utente. In questo caso bisogna ricordare che difficilmente tutti i programmi sulla cassetta di RE&C staranno su una sola facciata del disco su cui vengono trasferiti, perciò a un certo punto il trasferimento si bloccherà e il drive segnalerà errore per mancanza di spazio. Vi consigliamo quindi di utilizzare l'opzione 1 anche se volete trasferire tutti i programmi della cassetta.*

3 *- Questa opzione consnte di visionare la directory del disco.*

Il C64 al telefono

Qualche tempo fa ho acquistato un Commodore 64 con registratore. Qualche giorno fa ho saputo dell'esistenza di un adattatore telematico e vari modelli di modem costruiti appositamente per il mio computer. Prima di acquistare uno di questi accessori vorrei sapere se è possibile comunicare via telefono con altri computer pur non possedendo un drive.

Marco Migliavacca
Cuneo

Il collegamento del computer a un modem, e quindi ad una linea telefonica, è possibile, anche senza alcuna periferica o memoria di massa.
L'unica funzione del drive e del registratore è quella di permettere il caricamento del programma di gestione modem. Nel caso dell'adattatore telematico Commodore non è neppure necessario avere un software di gestione poiché questo si trova già su Rom all'interno dell'adattatore stesso e subito utilizzabile all'accensione del computer. I problemi per chi non possiede un drive cominciano a sorgere se l'utente desidera registrare in qualche modo i dati in arrivo o trasmettere dati preregistrati. Infatti, la maggior parte dei programmi di gestione modem, incluso quello interno dell'adattatore Commodore, non prevedono l'uso di un registratore a cassette. Per chi volesse provare a realizzare un software per un C64 con registratore e modem, proponiamo alcune nozioni di carattere generale su come utilizzare, mediante istruzioni Basic, la porta seriale RS-232 collegata a un modem.
La RS-232 viene aperta in questo modo:

OPEN lfn,2,0,"<registro di controllo><registro di comando>"

Lfn sta per logical file number, con lo stesso significato che ha nell'apertura dei file su disco, nastro, stampante eccetera.
Il registro di controllo ha il seguente significato:

Bit 7 - bit di stop (se 0 allora 1 bit di stop, se 1 allora 2 bit di stop)

Bit 6 e 5 - lunghezza della parola (00=8 bit, 01=7 bit, 10=6 bit, 11=5 bit)

Bit 4 - non utilizzato

Bit 3, 2, 1 e 0 - Baude rate (0110=300 baud)

Il registro di comando ha invece il seguente significato:

Bit 7, 6 e 5 - opzioni di parità (000 = disabilitata, 001 = parità dispari, 011 parità pari)

Bit 4 - duplex (0 = full duplex, 1 = half duplex)

Bit 4, 3, 2 e 1 - non utilizzati

Bit 0 - handshake

Per leggere e scrivere si utilizzano le normali istruzioni GET#,lfn oppure INPUT#,lfn e PRINT#,lfn. Ovviamente un programma di trasmissione e ricezione dati deve tenere conto di altri fattori come il buffer e il registro di stato.

La memoria nascosta

Perché non pubblicate una routine in L.m. per ampliare la Ram del Commodore 64, tale cioè che renda disponibile tutta la memoria da 2049 fino a 53247 per programmi in linguaggio macchina o basic compilati? Inoltre, è possibile utilizzare lo stesso sistema per rendere disponibile la memoria aggiunta solo alle variabili di un programma Basic?

Luigi Muzio
Sestri Levante (Sp)

Per quanto riguarda la routine in linguaggio macchina la soluzione è molto semplice. Infatti sono sufficienti tre istruzioni assembler per rendere disponibile la Ram che solitamente è nascosta <<sotto>> la Rom dell'interprete Basic. L'interprete Basic in pratica, consente la lettura, interpretazione e esecuzione del programma basic residente nelle locazioni da 2049 a 40959. È logico, quindi, che sostituendo alla Rom dell'interprete la Ram alternativa si impedisce l'esecuzione di qualsiasi programma basic non compilato.
Per quanto riguarda i programmi compilati il problema potrebbe non sussistere e il programma girare regolarmente dopo la chiamata alla micro-routine in linguaggio macchina.
Diciamo potrebbe perché in generale i programmi compilati fanno uso di alcune routine dell'interprete basic sulla Rom che la micro-routine esclude per fare posto alla Ram. Questa caratteristica, però, è strettamente legata al tipo di compilatore usato.
Per i programmi in linguaggio macchina la questione è analoga: affinché il programma funzioni dopo la chiamata della routine di espansione è necessario che il programma stesso non contenga alcun salto a routine dell'interprete Basic, ma faccia uso esclusivamente delle routine del sistema operativo, locato su Rom oltre la locazione 53247.
La routine di espansione è la seguente:

```
LDA #$36
STA  $01
RTS
```

In Basic:

```
10 FOR X=49152 TO 49157
20 READ A:POKE X,A
30 NEXT
40 DATA 169,54,133,1,96
```

Beach
Volley
LEADER
ocean